UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS

Models for multi-depot routing problems

" Documento Definitivo"

Doutoramento em Estatística e Investigação Operacional
Especialidade de Otimização

Daniel Rebelo dos Santos

Tese orientada por:
Professor Doutor Luís Eduardo Neves Gouveia

Documento especialmente elaborado para a obtenção do grau de doutor

2019

UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

Ciências
ULisboa

Models for multi-depot routing problems

Doutoramento em Estatística e Investigação Operacional

Especialidade de Otimização

Daniel Rebelo dos Santos

Tese orientada por:

Professor Doutor Luís Eduardo Neves Gouveia

Júri:

Presidente:

- Doutora Maria Eugénia Vasconcelos Captivo, Professora Catedrática, Faculdade de Ciências da Universidade de Lisboa

Vogais:

- Doutor Agostinho Miguel Mendes Agra, Professor Auxiliar, Departamento de Matemática da Universidade de Aveiro
- Doutor José Manuel Vasconcelos Valério de Carvalho, Professor Catedrático, Escola de Engenharia da Universidade do Minho
- Doutor José Manuel Pinto Paixão, Professor Catedrático, Faculdade de Ciências da Universidade de Lisboa
- Doutor Luís Eduardo Neves Gouveia, Professor Catedrático, Faculdade de Ciências da Universidade de Lisboa (Orientador)
- Doutora Maria Eugénia Vasconcelos Captivo, Professora Catedrática, Faculdade de Ciências da Universidade de Lisboa

Documento especialmente elaborado para a obtenção do grau de doutor

Esta dissertação foi financiada pela Universidade de Lisboa ao abrigo do Programa de Bolsas de Doutoramento da Universidade de Lisboa

2019

Agradecimentos

Em primeiro lugar queria agradecer à Raquel. Tudo o que faço, incluindo esta dissertação, é-lhe totalmente dedicado. Eu não seria a mesma pessoa e o meu percurso académico até este nível não teria existido se nunca tivesse tido o privilégio de a conhecer e de ser seu companheiro. O trabalho que aqui deixo foi grande parte fruto da sua paciência para me ouvir, para conversar comigo e, em especial na fase final, para me garantir as condições de trabalho ideais. Eu sei que achará que estou a exagerar, mas ambos sabemos que no fundo, e objetivamente, a Raquel terá não uma, mas duas dissertações de doutoramento de sua autoria, a sua e a minha.

Queria prestar um agradecimento muito especial e sentido ao meu orientador, o Professor Doutor Luís Eduardo Neves Gouveia, pela sua disponibilidade e entrega a esta dissertação a 100% e pelo vasto conhecimento que me passou. Foi, mais uma vez, uma grande honra ser orientado por um dos melhores do mundo na sua área que, apesar disso, sempre me tratou como se fôssemos normais colegas de trabalho. Olharei para todos os momentos passados no seu gabinete a escrever no quadro branco com grande saudade.

Gostaria de agradecer à minha mãe, Elvira, pelo seu apoio incondicional em todas as decisões que tomei, que tomo e que tomarei, e pela educação pessoal e académica que me proporcionou. Ter a certeza que em qualquer situação poderei sempre recorrer a ela é fundamental para mim. Agradeço também ao senhor Joaquim, pai da Raquel, por, apesar de não ser sua obrigação, assegurar constantemente o meu bem-estar. Um obrigado também à Maria Fernanda, avó da Raquel, e ao senhor Júlio, avô da Raquel, por me permitirem recorrer à sua ajuda sem nunca me rejeitarem nada.

I would like to thank Mario Ruthmair, for his availability in teaching me all the computational tools which I required for this dissertation and for never rejecting to help me in any situation, and Tolga Bektaş, for indirectly teaching me many amusing English expressions and for his important contribution to the work developed in this dissertation. Quero ainda agradecer a todos os meus professores, em especial àqueles que me permitiram participar em várias conferências que muito contribuíram para a minha evolução. Gostaria também de agradecer ao Michele, meu colega de gabinete durante grande parte desta dissertação, pela sua ajuda e pela sua companhia. Grazie mille, Michele.

Agradeço ainda a duas instituições importantes na realização desta dissertação. À SISCOG,

por me proporcionar as condições justas para que pudesse terminar a dissertação, e à Universidade de Lisboa, por financiar esta dissertação ao abrigo do Programa de Bolsas de Doutoramento da Universidade de Lisboa.

Finalmente, a todos os que de uma forma ou outra contribuíram para esta dissertação, nem que tenha sido apenas por me ouvirem ou por se mostrarem interessados no meu trabalho, o meu obrigado.

Abstract

In this dissertation we study two problems. In the first part of the dissertation we study the multi-depot routing problem. In the multi-depot routing problem we are given a set of depots and a set of clients and the objective is to find a set of routes with minimum total cost, one for each depot, such that each route starts and ends at the same depot and all clients are visited in one and only one route. The requirement that routes must start and end at the same depot is modeled by so-called path elimination constraints. We present a formulation which includes a newly developed set of multi-cut path elimination constraints and a branch-and-cut algorithm based on the new formulation that it is able to solve both asymmetric and symmetric instances with up to 300 clients and 60 depots. Additionally, we present other approaches to model path elimination constraints, including a formulation which provides linear programming relaxation values which are close to the optimal value in the instances tested.

In the second part of the dissertation we study the Hamiltonian p-median problem. In the Hamiltonian p-median we are given a set of nodes and the objective is to find p circuits with minimum total cost such that each node is in one and only circuit. We propose a formulation based on the concept of acting depot which attributes the role of artificial depot to p of the nodes. This formulation is a non-straightforward adaptation of the new model proposed for the multi-depot routing problem and it is based on a novel idea in which the standard arc variables are split into three cases depending on whether none or exactly one of its endpoints is an acting depot. We present a branch-and-cut algorithm based on the new formulation which is able to solve asymmetric instances with up to 171 nodes and symmetric instances with up to 100 nodes.

Keywords: multi-depot routing, Hamiltonian p-median, integer linear programming, projection, branch-and-cut

Resumo

Nesta dissertação estudamos dois problemas de otimização. Na primeira parte da dissertação abordamos o *multi-depot routing problem*. Dado um grafo cujos nodos são particionados num conjunto de depósitos e num conjunto de clientes, o objetivo do *multi-depot routing problem* é encontrar um conjunto de rotas, uma para cada depósito, tais que: (i) cada cliente é visitado numa e numa só rota; (ii) cada rota começa e termina no mesmo depósito; e (iii) o custo das rotas é o menor possível. Consideramos que o custo de uma rota é a soma dos custos dos arcos utilizados, sendo que nesta dissertação permitimos custos assimétricos, isto é, o custo de ir de A para B pode não ser o mesmo de ir de B para A. Assim sendo, os modelos que apresentamos são baseados em grafos orientados.

A condição (ii) de que cada rota tem de começar e terminar no mesmo depósito é geralmente modelada com recurso a restrições que na literatura se designam por *path elimination constraints*. Estas restrições garantem que qualquer caminho que ligue dois depósitos não é admissível. Nesta dissertação propomos uma formulação baseada num novo conjunto de *path elimination constraints* que podem ser vistas como restrições de corte num grafo com três níveis. Além do nível dos clientes, o grafo com três níveis tem um nível para os depósitos e outro para uma cópia de cada depósito, sendo que os arcos que entram no depósito no grafo original, entram na sua cópia no grafo com três níveis. As novas restrições resultam da projeção no espaço das usuais variáveis associadas a cada arco de um sistema de fluxos definido no grafo com três níveis que garante que uma unidade de fluxo é enviada de cada depósito para a sua cópia. Esta relação deriva do teorema de fluxo máximo/corte de capacidade mínima o que nos permite desenvolver um algoritmo de separação exato e eficiente para as novas restrições.

Com base na nova formulação apresentamos um algoritmo de *branch-and-cut* que faz uso da separação eficiente das novas *path elimination constraints* para resolver instâncias, tanto com custos assimétricos como com custos simétricos, com até 300 clientes e até 60 depósitos. O algorithm de *branch-and-cut* foi implementado em C++ e utiliza a *framework* do CPLEX que é um *software* que permite a resolução de problemas de otimização com base em modelos para problemas de programação inteira. O algoritmo de *branch-and-cut* incorpora técnicas para garantir a sua eficiência, como a limitação do número de desigualdades violadas adicionadas em cada iteração do algoritmo de planos de corte e a utilização de uma heurística simples, mas

eficaz, que produz soluções admissíveis com base na relaxação linear em cada nodo da árvore de pesquisa.

Nesta dissertação apresentamos também outras abordagens para modelar a restrição de que cada rota tem de começar e terminar no mesmo depósito. Alguns dos modelos apresentados têm por base um conjunto de variáveis que indicam se um dado cliente está ou não no circuito de um dado depósito. Com base em igualdades simples é possível mostrar que estas variáveis estão associadas às variáveis de fluxo do sistema de fluxos definido no grafo com três níveis, logo, é possível estabelecer uma comparação com os modelos apresentados anteriormente, em particular as novas *path elimination constraints* propostas. Apresentamos ainda outro modelo baseado em dois sistemas de fluxo, um que garante que uma unidade de fluxo é enviada de cada depósito para cada cliente que esteja no seu circuito e outro que garante o contrário, de tal forma que, em conjunto, é garantida a existência de um circuito para cada depósito. Mostramos também que estes dois sistemas de fluxo podem ser relacionados com o sistema de fluxos do grafo com três níveis e, recorrendo a técnicas de projeção, mostramos que é possível definir um modelo equivalente que não necessita dos sistemas de fluxo duplos e que é mais fácil de utilizar na prática. Por fim, apresentamos um conjunto de resultados computacionais para avaliar a qualidade dos valores da relaxação linear dos vários modelos apresentados e, em particular, mostramos que o modelo que resulta dos sistemas de fluxo duplos é um modelo cujo valor da relaxação linear está perto do valor ótimo nas instâncias testadas.

Na segunda parte da dissertação estudamos o *Hamiltonian p-median problem*. Para este problema é-nos dado um grafo (que mais uma vez assumimos orientado) e uma função de custo associada aos arcos do grafo. O objetivo do *Hamiltonian p-median problem* é encontrar p circuitos tais que: (i) cada nodo do grafo está num e num só circuito; e (ii) o custo dos p circuitos é o menor possível. Começamos por apresentar um modelo genérico para o problema que é baseado no conceito de depósito artificial, isto é, p dos nodos do grafo são escolhidos para atuarem como um depósito artificial e, por conseguinte, os restantes nodos são clientes artificiais. Com base neste conceito estabelecemos a ligação ao *multi-depot routing problem* e apresentamos um novo modelo que é uma adaptação do modelo baseado nas novas restrições de corte associadas ao grafo de três níveis. Contudo, observamos que este modelo possui duas desvantagens. Em primeiro lugar, não cremos que exista um algoritmo de separação eficiente para as restrições de corte associadas ao grafo de três níveis adaptadas para o contexto do *Hamiltonian p-median problem*. Em segundo lugar, a utilização do conceito de depósito artificial introduz problemas de simetria dado que um circuito pode ser representado de forma equivalente tantas vezes quanto o número de nodos que o compõem. Neste modelo não é possível resolver este problema de forma intuitiva.

Assim, apresentamos um novo modelo para o *Hamiltonian p-median problem* baseado na ideia de dividir as variáveis associadas aos arcos em três conjuntos distintos consoante nenhum

dos seus extremos ou exatamente um dos seus extremos é um depósito artificial. A nova formulação permite-nos na mesma adaptar (de forma não trivial) os modelos propostos para o *multi-depot routing problem* e, mais importantemente, colmatar as duas desvantagens do primeiro modelo proposto. Mais concretamente, é possível definir um conjunto de restrições semelhantes às restrições de corte associadas ao grafo de três níveis apresentadas para o *multi-depot routing problem* de tal forma que: (i) a sua separação pode ser feita em tempo polinomial; e (ii) podem ser adaptadas para lidar com os problemas de simetria inerentes à utilização do conceito de depósito artificial.

Para terminar, apresentamos um segundo algoritmo de *branch-and-cut*, desta feita para o *Hamiltonian p-median problem*, que utiliza o mesmo tipo de técnicas que o apresentado para o *multi-depot routing problem*. Este algoritmo de *branch-and-cut* permite resolver instâncias com custos assimétricos com até 171 nodos e de instâncias simétricas com até 100 nodos. Fazemos ainda uma comparação com um terceiro algoritmo de *branch-and-cut* baseado na adaptação de uma formulação da literatura. Os resultados mostram que o algoritmo de *branch-and-cut* baseado na nova formulação é bastante mais eficiente, em média, do que este terceiro algoritmo. Por fim, mostramos ainda um conjunto de resultados que permitem comparar a nossa abordagem com abordagens da literatura que resolvem uma variante do *Hamiltonian p-median problem* em que circuitos com apenas dois nodos não são permitidos.

Palavras-chave: rotas com múltiplos depósitos, p-mediana Hamiltoniana, programação linear inteira, projeção, *branch-and-cut*

Contents

List of Figures

List of Tables

Introduction

The field of Operations Research is a thriving mathematics discipline with the main purpose of proposing solution methods for optimization problems. One of the most important and widely studied type of problems are routing problems. Routing problems usually stem from the area of distribution logistics where one wishes to find a set of routes that visit given locations while simultaneously minimizing some objective (e.g., operational costs). Often a route starts and ends at a specific location denoted by depot and some companies may have several depots from where their distribution vehicles depart.

In this dissertation we study routing problems in which there exists more than one depot or, in other words, multi-depot routing problems. The dissertation is divided into two parts, each one corresponding to a different optimization problem. In the first part of the dissertation we study the multi-depot routing problem. In this optimization problem we are given a set of depots and a set of locations that need to be visited. The objective is to find a set of routes, one for each depot, such that: (i) each location is visited in one and only one route; (ii) each route starts and ends at the same depot; and (iii) a given cost function is minimized. In the second part of the dissertation we study the Hamiltonian p-median problem where, given a set of locations, we must find p circuits such that: (i) each location is in one and only one circuit; and (ii) a given cost function is minimized. Albeit different problems, we show in this dissertation that the multi-depot routing problem can be seen as a particular case of the Hamiltonian p-median problem.

We strongly believe that the additional condition in multi-depot routing problems that routes must start and end at the same depot has not been given sufficient attention in the literature. Additionally, we believe that the Hamiltonian p-median is lacking recent algorithmic development given that most works in the literature address a variant of the problem. These two points justify the work developed within the scope of this Ph.D. dissertation.

This dissertation has three distinct aims. The first aim is to present a number of different mathematical models for the multi-depot routing problem focusing, in particular, on modeling the restriction that routes must start and end at the same depot. The second aim is to provide an efficient solution method which can provide good quality (hopefully, the best possible) solutions for the multi-depot routing problem based on the theoretical models proposed. The third and

final aim is to establish a link between the multi-depot routing problem and the Hamiltonian p-median problem which allows us to (not necessarily in a straightforward way) adapt both the models and the solution methods proposed for the former to the latter.

This dissertation is organized in the following way. Part I is divided into four chapters. In Chapter 1 we formally introduce the multi-depot routing problem. In particular, we present notation used throughout the first part of the dissertation and a generic model for the problem. In Chapter 2 we propose a model for the multi-depot routing problem which includes newly developed constraints that guarantee that each route starts and ends at the same depot. In Chapter 3 we propose a branch-and-cut algorithm based on the model of Chapter 2 and a computational experiment to evaluate its performance. Finally, in Chapter 4 we present additional models for the multi-depot routing problem, once again focusing on modeling the condition that routes must start and end at the same depot.

In Part II of the dissertation we study the Hamiltonian p-median problem. In Chapter 5 we introduce the problem and notation as well as two different generic models. One of these models establishes a link to the multi-depot routing problem by introducing the concept of acting depot which attributes the role of artificial depots to p of the locations in the problem. In Chapter 6 we use this concept and present a new model for the Hamiltonian p-median problem which is a non-straightforward adaptation of the model of Chapter 2. Finally, in Chapter 7 we present another branch-and-cut algorithm to solve the Hamiltonian p-median problem and test its effectiveness.

Part I

The multi-depot routing problem

Chapter 1

Introducing the multi-depot routing problem

Contents

1.1 Introduction

In this chapter we introduce the problem studied in the first part of this dissertation, the multi-depot routing problem (see, e.g., Laporte, Nobert & Arpin 1984, Laporte, Nobert & Taillefer 1988, Bektaş, Gouveia & Santos 2017). This problem arises in distribution logistics where vehicles located at a set of depots are used to service a set of clients. The multi-depot routing problem is an extension of more traditional routing problems which consider only a single depot with a single vehicle, such as the traveling salesman problem (see, e.g., Lawler, Lenstra, Kan & Shmoys 1985, Applegate, Bixby, Chvátal & Cook 2006), or a single depot with multiple vehicles, which includes the multiple traveling salesman problem (see, e.g, Gavish & Graves 1978, Bektaş 2006) and the vehicle routing problem and its several variants (see, e.g., Toth & Vigo 2014). In the multi-depot routing problem one assumes the existence of multiple depots, each one with one or more vehicles. In this dissertation we will consider that there exist multiple depots with a single vehicle in each but we will also provide some insight on how the discussion can be extended to the case in which there are multiple vehicles in each depot. In addition, given that there exist multiple depots, it is interesting to consider also the variant that allows choosing which depots to use. We also show how to adapt the discussion for this variant. Our main focus,

however, will be the case in which each depot has a single vehicle that must be used in a circuit since this case ties in perfectly with the problem studied in the second part of this dissertation.

Most of the recent routing problems in the literature also take into account other restrictions resulting from real-life applications (see, e.g., Toth & Vigo 2014, for some examples regarding the vehicle routing problem). For instance, the use of multiple vehicles is usually required whenever the vehicles have a maximum service capacity, time windows for servicing a particular client are also frequently considered, environmental issues are also important nowadays, specially because of the growing use of electric vehicles or simply to minimize the pollution, etc. This dissertation is focused on studying multi-depot routing problems in their most basic setting, so we will not add any additional restrictions. The reason for this choice is that the multi-depot routing problem in its most basic setting has not been studied in detail, specially in what concerns the additional constraint of requiring that vehicles return to their original departure depot, and, therefore, adding any additional restrictions would deviate from our original purpose. Nevertheless, the study in this dissertation may serve as a base for future adaptations that consider such additional restrictions on multi-depot routing problems.

The aim in most routing problems is usually to minimize a cost function that is related to the routes that each vehicle must follow. The most often used cost function is the total distance traveled by all the vehicles. However, in this dissertation we are going to assume that the costs can be more general. In particular, the cost of going from a point A to a point B may not be the same cost of going from B to A. For instance, due to the topology of the network, one of the directions may be non-existent or it may take longer to traverse. This means that the cost function is an asymmetric cost function. Consequently, it is important to distinguish whether we are going from A to B or from B to A and, therefore, we will base our models on directed graphs.

The two types of restrictions that must be satisfied in the multi-depot routing problem are: (i) each client is serviced in one and only one route; and (ii) each route contains exactly one depot. The latter set of constraints requires further elaboration. Firstly, a route that does not contain a depot, is one of client nodes alone and disconnected from the depots. Inequalities that prevent the formation of such routes are known as subtour elimination constraints. This type of constraints has been widely studied in the context of single-depot routing problems and their adaptation to multi-depot routing problems is usually straightforward. Secondly, if a route contains two or more depots, this implies that there exists a vehicle traveling on a path between two different depots. Since we assume that a vehicle needs to return to its original departure depot after the travel, any such paths that exist between different depots are not acceptable. Inequalities that disallow unfeasible paths are called path elimination constraints. Unfeasible paths between different depots are non-existent in single-depot routing problems and, therefore, the study of path elimination constraints is not as vast as the study of subtour elimination con-

straints. For this reason, one of the main topics of this part of the dissertation is the study of path elimination constraints. Note that the study of these constraints is relevant for any problem where multiple depots exist including, for example, location-routing problems (see, e.g., Laporte, Nobert & Arpin 1986, Laporte et al. 1988, Belenguer, Benavent, Prins, Prodhon & Wolfler Calvo 2011, Albareda-Sambola 2015) and other variants (see, e.g., Laporte et al. 1984, Bektaş 2012, Benavent & Martínez-Sykora 2013, Fernández & Rodríguez-Pereira 2016, Sundar & Rathinam 2017).

The purpose of this part of the dissertation is two-fold. First, we propose a new formulation for the multi-depot routing problem and devise an efficient and effective branch-and-cut algorithm based on this formulation. More precisely, we present a formulation in the space of the standard arc variables usually used in routing problems that includes a set of subtour elimination constraints, a set of newly developed path elimination constraints and other valid inequalities. The newly developed path elimination constraints are multi-cut constraints and are one of the most important contributions of this dissertation. The multi-cut constraints result from the projection in the space of the arc variables of a compact three-index variable based formulation, which can be interpreted as a network flow formulation. For the branch-and-cut algorithm, we start by explaining what a modern branch-and-cut algorithm implementation in a commercial solver consists of. Then, we present some state-of-the-art techniques which we use in our branch-and-cut algorithm, such as a basic but fast and effective heuristic and heuristic separation algorithms for the inequalities in use. The branch-and-cut algorithm is then used to solve a number of benchmark instances and some randomly generated instances.

The second purpose of this part of the dissertation is to provide a comparative study, both in theory and in practice, of several modeling approaches for path elimination constraints. Several of the sets of path elimination constraints proposed are based on modeling techniques similar to the ones used in the precedence constrained (asymmetric) traveling salesman problem (see, e.g., Balas, Fischetti & Pulleyblank 1995, Gouveia & Pires 1999, 2001, Gouveia & Pesneau 2006, Gouveia, Pesneau, Ruthmair & Santos 2018), namely in what concerns the use of variables that are similar to the so-called precedence variables and that, in the multi-depot routing problem setting, are interpreted as variables which assign clients to depots. These will be theoretically compared to the compact three-index variable based formulations and the formulation using the new multi-cut constraints. In addition, we use double network flow formulations similar to the ones proposed by Wong (1980) for the traveling salesman problem to derive a new formulation which implies all of the proposed subtour elimination constraints and path elimination constraints presented in this part of the dissertation. This formulation is another important contribution of this dissertation and we will show that it provides linear programming relaxation values which are close to the optimal integer solution value in the instances tested.

This chapter is divided into two sections. In Section 1.2 we formally define the multi-depot

routing problem and present notation to ease the writing of some mathematical expressions. Afterwards, in Section 1.3, we present a generic integer linear programming formulation for the multi-depot routing problem.

1.2 Definitions and notation

We define the multi-depot routing problem on a directed graph $G = (V, A)$. The set of nodes $V = \{1, 2, \ldots, n\}$ is partitioned into two sets D and C, where D is the set of nodes which are depots, each one a departure point of a vehicle with unlimited service capacity, and C is the set of nodes which are clients. Since traveling directly between two depots is forbidden, there exist no arcs between depot nodes, however, we will assume that all arcs between depot nodes and client nodes exist and that all arcs between client nodes exist, hence, the set of arcs A is complete apart from the non-existing arcs between depots, that is, $A = \{(i, j) : i \in C \text{ or } j \in C,\ i \neq j\}$. Note that the assumption of a (almost) complete graph does not lose any generality as the ensuing exposition can easily be adapted to incomplete graphs by simply not considering the pairs (i, j) such that $(i, j) \notin A$ in any mathematical expression. The arc set can be partitioned as follows: let $A^C = \{(i, j) \in A : i, j \in C,\ i \neq j\}$ be the set of arcs between client nodes; for all $d \in D$ let $A^d_O = \{(d, i) \in A : i \in C\}$ be the set of arcs outgoing depot d; and for all $d \in D$ let $A^d_I = \{(i, d) \in A : i \in C\}$ be the set of arcs ingoing depot d. Finally, we consider a general non-negative cost function c associated with each existing arc.

Later in this part of the dissertation we will briefly consider two variants of the multi-depot routing problem. In one case we will assume that there is a fixed number k_d of vehicles available at each depot $d \in D$ and that at least one of them must be used. In another case we will consider the possibility that the nodes of D are seen as potential depot locations and, consequently, they may not need to be used. Mixing these two variants is possible, in which case we can decide how many vehicles from the k_d available are to be used for each depot $d \in D$ including choosing to not use any. We believe that starting with the more general case would be counterproductive since, as we have mentioned, we wish to study the multi-depot routing problem in its most basic setting, particularly in what concerns path elimination constraints, and, therefore, it is much easier to explain the intuitiveness of the modeling techniques in the simpler setting and later show how to adapt the discussion to a more general case. For this reason, and with the exception of Section 2.6, we will assume that a vehicle exists at every depot of D and that the vehicle must be used.

The objective of the multi-depot routing problem is then to find a minimum cost set of $|D|$ circuits such that each circuit departs from and ends at the same depot and each client of C is in one and only one circuit. Figure 1.1 shows an example of a feasible solution of a multi-depot routing problem where the depots $D = \{1, 2\}$ are represented as squares and the clients

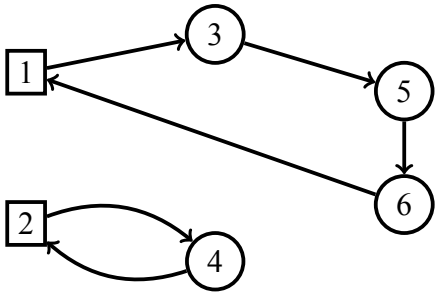

Figure 1.1: An example of a feasible solution of a multi-depot routing problem

$C = \{3, 4, 5, 6\}$ are represented as circles. From depot 1 there is a vehicle that departs and services client 3, then client 5 and then client 6, and returns to the depot 1. From depot 2 there is another vehicle that services only client 4 before coming back to the depot from where it originally left.

In order to simplify the mathematical expressions we will be using the following notation:

- For any general one-index variable u, we write $u(S) = \sum_{i \in S} u_i$;
- For any general two-index variable v in which both indexes are subscripts, we write $v(S) = \sum_{i,j \in S} v_{ij}$ and $v(S_1, S_2) = \sum_{i \in S_1, j \in S_2} v_{ij}$;
- For any general two-index variable w with one subscript index and one superscript index, we write $w_{S_1}^{S_2} = \sum_{i \in S_1, j \in S_2} w_i^j$;
- For any general three-index variable z with two subscript indexes and one superscript index, we write $z^k(S) = \sum_{i,j \in S} z_{ij}^k$ and $z^k(S_1, S_2) = \sum_{i \in S_1, j \in S_2} z_{ij}^k$;
- In the expressions above, for any singleton set $\{i\}$ we write i instead of $\{i\}$;
- Finally, we define $S' = C \setminus S$ for any client subset $S \subseteq C$.

1.3 A generic model

The multi-depot routing problem can be modeled as an integer linear programming problem. Consider a set of binary variables $x_{ij} = 1$ if arc $(i, j) \in A$ is used in any of the circuits, and $x_{ij} = 0$ otherwise. We will start by presenting a model using the x variables in which some sets of constraints are defined in a generic way.

$$\text{Minimize} \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{1.1}$$

$$\text{subject to:} \sum_{j \in C} x_{dj} = 1 \qquad \forall d \in D \tag{1.2}$$

$$\sum_{j \in C} x_{jd} = 1 \qquad \forall d \in D \qquad (1.3)$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in C \qquad (1.4)$$

$$\sum_{j \in V} x_{ji} = 1 \qquad \forall i \in C \qquad (1.5)$$

$\{(i,j) \in A : x_{ij} = 1\}$ contains no circuit with zero depots (1.6)

$\{(i,j) \in A : x_{ij} = 1\}$ contains no circuit with two or more depots (1.7)

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \qquad (1.8)$$

The model represented by (1.1)–(1.8) is a valid generic integer linear programming model for the multi-depot routing problem. Notice that the system of inequalities (1.2)–(1.5) and (1.8) is the integer linear programming formulation of the assignment problem if we reintroduce the arcs between depots with a sufficiently high cost (see, e.g., Wolsey 1998) and, for this reason, we will from now on call it the assignment relaxation.

Constraints (1.2) and (1.4) ensure that any depot and client must have an outdegree of 1, respectively, whereas (1.3) and (1.5) ensure that the indegree is also 1. The only way this can be satisfied is if the arcs chosen to be used form a set of disjoint circuits, that is, any feasible solution of the assignment relaxation defines a set of disjoint circuits on graph G. However, a feasible solution for the assignment relaxation may not be a feasible solution for the multi-depot routing problem. On the one hand, every client is part of exactly one circuit but, on the other hand, it is not guaranteed that all circuits have exactly one depot. For this reason we need additional sets of constraints. More precisely, constraints (1.6) ensure that circuits with zero depots, or equivalently client-only circuits, cannot exist, while constraints (1.7) guarantee that circuits with more than one depot cannot exist.

Constraints (1.6) are common to all routing problems and are usually called subtour elimination constraints. For this reason, they have been widely studied in the context of other routing problems and they will not be the main focus of this dissertation. Obviously they are still fundamental in any formulation for the multi-depot routing problem and so we will discuss an effective way of modeling them in Section 2.2 and also refer to adequate related literature. Constraints (1.7) are specific to the multi-depot routing setting and are called path elimination constraints. In this case, they ensure that a path between two different depots does not exist and, thus, prevent the existence of circuits with more than one depot. One of the main objectives of this part of the dissertation is the study of such constraints, since in our opinion there are not enough comparisons in the literature between different ways of modeling path elimination constraints in the multi-depot routing problem setting.

In the next chapter we present a valid integer linear programming formulation, defined in the space of the x variables and based on this generic model, for the multi-depot routing problem.

Chapter 2

A new formulation for the multi-depot routing problem

Contents

2.1 Introduction

In this chapter we propose a new formulation for the multi-depot routing problem defined in the space of the arc variables x. More precisely, we use the generic formulation presented in Section 1.3 with the addition of sets of constraints in the space of the x variables to model the generic subtour elimination constraints (1.6) and the generic path elimination constraints (1.7).

In Section 2.2 we discuss subtour elimination constraints in general and provide references to adequate literature. Recall that subtour elimination constraints have been extensively studied in more traditional routing problems and so they are not the main focus of this part of the dissertation. For the purpose of defining the new formulation we will present, in particular, an effective set of subtour elimination constraints.

In Section 2.3 we define a compact system of inequalities which models path elimination constraints based on a set of variables which assign arcs to specific circuits, and discuss some of its properties. In particular, we show that this system of inequalities can be interpreted as a network flow formulation in an adequate graph.

In Section 2.4 we present a new set of path elimination constraints defined in the space of the x variables which will be used in the new formulation. These newly developed path elimination constraints are multi-cut constraints which are related to the compact systems of inequalities of Section 2.3. This relationship will allow us to derive a generalization of the multi-cut constraints, as well as devise separation algorithms for the basic multi-cut constraints and their generalizations.

In Section 2.5 we present a set of path elimination constraints from the literature and compare them to the multi-cut constraints of Section 2.4. Then, in Section 2.6 we show how to adapt the new formulation to related problems involving multiple depots and, finally, we finish in Section 2.7 with some concluding remarks.

2.2 Subtour elimination constraints

Subtour elimination constraints have been widely studied in the literature in the context of routing problems. There exist a number of recent surveys (see, e.g., Öncan, Altınel & Laporte 2009, Godinho, Gouveia & Pesneau 2011, Roberti & Toth 2012) on subtour elimination constraints for the (asymmetric) traveling salesman problem that can be straightforwardly adapted for eliminating client-only circuits in other related problems. These surveys also include comparisons of compact and non-compact formulations in practice and in theory. Therefore, we will not perform a comparative study of these subtour elimination constraints in the multi-depot routing problem setting and, instead, we will use one of the most effective set of subtour elimination constraints in general due to Dantzig, Fulkerson & Johnson (1954), adapted to the context of the

multi-depot routing problem.

Consider a client set $S \subset C$. Any circuit spanning all nodes of S necessarily uses $|S|$ arcs between nodes of S, hence, the following subtour elimination constraints eliminate these unfeasible client-only circuits by limiting the number of arcs that can be used in any set S in these conditions:

$$x(S) \leq |S| - 1 \qquad \forall S \subset C : \ 2 \leq |S| \leq |C| - |D|. \tag{2.1}$$

Note that, due to the depot outdegree constraints (1.2) and the fact that there exist no arcs between depots, an upper limit on $|S|$ is given by $|C| - |D|$. By using the client indegree constraints (1.5), constraints (2.1) can be equivalently written in the following cut form:

$$x(D \cup S', S) \geq 1 \qquad \forall S \subset C : \ 2 \leq |S| \leq |C| - |D|. \tag{2.2}$$

This second representation is useful in order to derive a separation algorithm, which is fundamental since these constraints are in exponential number. In Section 3.3.1 we will present a polynomial-time exact separation algorithm for the subtour elimination constraints (2.2) which is based on max-flow/min-cut computations in an adequate graph and which is an adaptation of known separation algorithms in the context of other routing problems.

2.3 Path elimination constraints based on arc-depot assignment variables

In this section we show how to model path elimination constraints by using a set of arc-depot assignment variables $z^d_{ij} = 1$ if arc $(i, j) \in A$ is used in the circuit of depot $d \in D$, and $z^d_{ij} = 0$ otherwise. We start by presenting a compact system of inequalities in Section 2.3.1, which is similar to ones used in other related problems (see, e.g., Albareda-Sambola, Díaz & Fernández 2005, Bektaş 2012, Fernández & Rodríguez-Pereira 2016, Hill & Voß 2016).

In Section 2.3.2 we show that this compact system of inequalities can be interpreted as a network flow model in an adequate graph, which will be fundamental in Section 2.4 when we relate this system of inequalities to a set of constraints in the space of the x variables. In Section 2.3.3 we show how to strengthen the linear programming relaxation of the base compact system of inequalities, and then further strengthen the resulting system of inequalities, by using arguments based on the disjointness of the circuits.

Finally, in Section 2.3.4 we present a property of the systems of inequalities based on the arc-depot assignment variables which is related to the linear programming relaxation value for symmetric cost instances.

2.3.1 A base model in the space of the x and the z variables

In this section we define a compact system of inequalities which models path elimination constraints by using the arc-depot assignment variables. First, notice that an arc with an endpoint in a depot $d \in D$ cannot be used in the circuit of another depot $d' \in D \setminus \{d\}$ or else we could have a path between depots d and d', hence, we define variables z_{ij}^d, for any $d \in D$, only for arcs $(i,j) \in A^C \cup A_O^d \cup A_I^d$. Consider, then, the following system of inequalities, which we will denote by 3I:

$$\sum_{j \in C} z_{dj}^d = 1 \qquad \forall d \in D \tag{2.3}$$

$$\sum_{j \in C} z_{jd}^d = 1 \qquad \forall d \in D \tag{2.4}$$

$$\sum_{j \in \{d\} \cup C} z_{ji}^d = \sum_{j \in \{d\} \cup C} z_{ij}^d \qquad \forall d \in D,\ \forall i \in C \tag{2.5}$$

$$z_{ij}^d \leq x_{ij} \qquad \forall d \in D,\ \forall (i,j) \in A^C \cup A_O^d \cup A_I^d \tag{2.6}$$

$$z_{ij}^d \in \{0,1\} \qquad \forall d \in D,\ \forall (i,j) \in A^C \cup A_O^d \cup A_I^d. \tag{2.7}$$

Constraints (2.3) and (2.4) state that for each depot $d \in D$ there must be an arc outgoing d and an arc ingoing d which is used in the circuit of depot d, respectively. Constraints (2.5) ensure that the outdegree and the indegree of each client node is the same for every circuit. In particular, if the indegree of a client $i \in C$ in the circuit of a given depot $d \in D$ is 1, then its outdegree must also be 1. Otherwise, if the indegree of i in the circuit of depot d is 0, then its outdegree must also be 0. Finally, constraints (2.6) link the z and the x variables by ensuring that if an arc is used in a given circuit then the corresponding x variable for that arc must be equal to 1. Conversely, if an arc is not used, then it cannot be used in any circuit. Intuitively, the 3I system guarantees that there must exist a path starting at each depot $d \in D$ and ending at the same depot d, hence, it prevents the existence of unfeasible paths between depots. We will look into this interpretation in more detail in Section 2.3.2.

To see that the 3I system models path elimination constraints, consider an unfeasible path $(d_1, i_1, i_2, \ldots, i_k, d_2)$ in which $d_1, d_2 \in D,\ d_1 \neq d_2$ and $i_1, i_2, \ldots, i_k \in C$. From constraints (2.3) we have that $z_{d_1 i_1}^{d_1} = 1$, thus, from using constraints (2.5) in succession for each node $i_1, \ldots, i_k$, we get $z_{i_{k-1} i_k}^{d_1} = 1$. However, the arc outgoing client i_k goes to depot d_2 which is impossible because variable $z_{i_k d_2}^{d_1}$ does not exist.

Despite being a compact system of inequalities, the 3I system can be difficult to use in practice for two reasons. Firstly, each arc between client nodes is replicated as many times as the number of depots in order to create the arc-depot assignment variables and, thus, the number of existing z variables can become significantly large as the number of depots increases. Secondly, the number of linking constraints between the z and the x variables (2.6) is also significantly

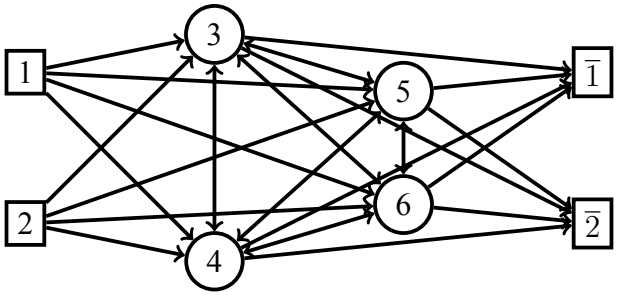

Figure 2.1: An example of the 3-layered graph

large since one constraint exists for each z variable. Regarding the second issue we will show in Section 2.3.3 that we can strengthen this compact system of inequalities in a way that uses considerably fewer linking constraints between the z and the x variables and define a system of inequalities which provides an at least as good corresponding linear programming relaxation value than the 3I system.

2.3.2 A network flow interpretation of the base model

In this section we give an important interpretation of the 3I system defined in Section 2.3.1 as a network flow model. This interpretation is not necessary in order to use the 3I system in practice, however, it is important in order to establish a relationship between the 3I system and a set of constraints in the space of the x variables, as we will see in Section 2.4.

For the network flow model interpretation, we require a specific graph, which we denote by 3-layered graph. In this graph, the set of depots D is replicated into set $\overline{D}$, with each node $\overline{d} \in \overline{D}$ being the copy of the original depot $d \in D$. With respect to the arc set defined in the new graph, the only difference is that arcs entering node d in the original graph now enter node $\overline{d}$. To the best of our knowledge, this 3-layered graph approach was first proposed by Albareda-Sambola et al. (2005) and later explored by Bektaş (2012). Figure 2.1 shows an example of the 3-layered graph for a multi-depot routing problem in which $D = \{1, 2\}$ and $C = \{3, 4, 5, 6\}$.

The 3I system can be viewed as a network flow model in the 3-layered graph which guarantees that 1 unit of flow is sent from each node $d \in D$ to its copy $\overline{d} \in \overline{D}$, where the z_{ij}^d variables are re-interpreted as indicating whether or not arc $(i, j) \in A^C \cup A_O^d \cup A_I^d$ is used to send one unit of flow from node d to its copy $\overline{d}$. Alternatively, observe that the condition that 1 unit of flow must be sent from each depot to its copy is equivalent to stating that a path must exist from each depot to its copy. In addition, given the network flow (or path) model interpretation, the integrality requirement on the z variables can be relaxed to $z_{ij}^d \geq 0$.

In the example of Figure 2.1, the 3I system ensures that one unit of flow is sent from node 1 to node $\overline{1}$ and from node 2 to node $\overline{2}$ in the 3-layered graph, or, alternatively, that two paths must exist, one going from node 1 to node $\overline{1}$ and another from node 2 to node $\overline{2}$.

2.3.3 Strengthening the base model

As we mentioned at the end of Section 2.3.1, the 3I system contains a large number of linking constraints $z_{ij}^d \leq x_{ij}$ (2.6). However, observe that any arc $(i,j) \in A^C$ can only be used in at most one circuit in any feasible solution of the multi-depot routing problem given the requirement that the circuits for each depot are disjoint. Consider then the following compact system of inequalities which dominates the 3I system and which we denote by 3I$^+$:

$$\sum_{j \in C} z_{dj}^d = 1 \qquad \forall d \in D \tag{2.3}$$

$$\sum_{j \in C} z_{jd}^d = 1 \qquad \forall d \in D \tag{2.4}$$

$$\sum_{j \in \{d\} \cup C} z_{ji}^d = \sum_{j \in \{d\} \cup C} z_{ij}^d \qquad \forall d \in D,\ \forall i \in C \tag{2.5}$$

$$\sum_{d \in D} z_{ij}^d \leq x_{ij} \qquad \forall (i,j) \in A^C \tag{2.8}$$

$$z_{ij}^d \leq x_{ij} \qquad \forall d \in D,\ \forall (i,j) \in A_O^d \cup A_I^d \tag{2.9}$$

$$z_{ij}^d \in \{0,1\} \qquad \forall d \in D,\ \forall (i,j) \in A^C \cup A_O^d \cup A_I^d. \tag{2.7}$$

The number of linking constraints between the z and the x variables (2.8)–(2.9) in the 3I$^+$ system is substantially lower than the number of the linking constraints (2.6) of the 3I system, since in the 3I system, as we mentioned before, there is one linking constraint for each z variable and in the 3I$^+$ system the linking constraints for arcs $(i,j) \in A^C$ are aggregated for all depots. Therefore, the 3I$^+$ system is preferable to use to model path elimination constraints than the 3I system for two reasons. Firstly, the number of variables is unchanged but the number of constraints is reduced. Secondly, the linking constraints (2.8)–(2.9) imply the weaker linking constraints (2.6), hence, the 3I$^+$ system is stronger than the 3I system.

The discussion of Section 2.3.2 can also be applied to the 3I$^+$ system, that is, this system of inequalities can also be interpreted as a network flow model in the 3-layered graph with additional constraints. More precisely, for each subset $D' \subseteq D$ the 3I$^+$ system ensures that $|D'|$ units of flow are sent from the depots in D' to their copies in the 3-layered graph and that these flows are disjoint. Equivalently, the 3I$^+$ system guarantees that there exist $|D'|$ disjoint paths going from the depots in D' to their copies. Given these interpretations, we can once again relax the integrality requirement on the z variables to $z_{ij}^d \geq 0$.

Observe that we can further improve the 3I$^+$ system by noticing that equality holds in both of the linking constraints (2.8) and (2.9). In the case of the former, note that if an arc $(i,j) \in A^C$ is used then it needs to be used in one and only one of the circuits. In the latter, if an arc with an endpoint in a depot $d \in D$ is used then it needs to be used in the circuit of the same depot. Consequently, we can replace the linking constraints (2.8) and (2.9) by the following

ones, respectively:

$$\sum_{d \in D} z_{ij}^d = x_{ij} \qquad \forall (i,j) \in A^C \tag{2.10}$$

$$z_{ij}^d = x_{ij} \qquad \forall d \in D,\ \forall (i,j) \in A_O^d \cup A_I^d. \tag{2.11}$$

We denote by 3I^{++} the system of inequalities 3I^{+} in which we replace the linking constraints (2.8)–(2.9) by constraints (2.10)–(2.11). When the objective function only depends on the x variables, we can prove that the linear programming relaxation value is the same regardless of whether we use equalities or inequalities in the linking constraints. We start by proving that the linking constraints (2.9) and (2.11) are equivalent and then we show that the 3I^{+} and the 3I^{++} systems provide the same linear programming relaxation value.

Proposition 1. *Consider a solution (x', z') that satisfies the depot degree constraints* (1.2) *and* (1.3) *for the x variables and constraints* (2.3) *and* (2.4) *for the z variables. Then, (x', z') satisfies the linking constraints* (2.9) *if and only if it satisfies the linking constraints* (2.11)*.*

Proof. Clearly if (x', z') satisfies constraints (2.11), then it also satisfies constraints (2.9). Conversely, consider a constraint (2.9) associated with a depot $d \in D$ and suppose that there exists a client $j \in C$ such that, for the arc $(d, j) \in A_O^d$, we have $z_{dj}^d < x_{dj}$. Then, we obtain $1 = \sum_{i \in C} z_{di}^d < \sum_{i \in C} x_{di} = 1$, where the first equality is given by constraints (2.3) and the second equality is given by the depot outdegree constraints (1.2), which is a contradiction. For the arcs of A_I^d, we can use a similar reasoning. □

Proposition 2. *Consider two models, model A comprised of the assignment relaxation constraints* (1.2)–(1.5)*, the domain constraints for the x variables* (1.8) *and the constraints of the 3I^{+} system, and model B comprised of the assignment relaxation constraints* (1.2)–(1.5)*, the domain constraints for the x variables* (1.8) *and the constraints of the 3I^{++} system. Given an objective function which only depends on the x variables, both models provide the same linear programming relaxation value.*

Proof. The idea of this proof is to show that we can construct a solution of the linear programming relaxation of model B based on a solution of the linear programming relaxation of model A where the part on the x variables is the same and, thus, the difference is only on the part on the z variables. For this reason, both solutions will have the same cost since we assume that the objective function only depends on the x variables.

Let (x', z') be a solution of the linear programming relaxation of model A and, for all arcs $(i,j) \in A^C$, define $\epsilon_{ij} = x'_{ij} - \sum_{d \in D} z'^d_{ij}$. We start by proving that $\epsilon(i, C) = \epsilon(C, i)$ for any $i \in C$. First notice that from the client outdegree constraints (1.4) we have that:

$$x'(i, C) = x'(C, i) + x'(D, i) - x'(i, D).$$

Additionally, from the flow conservation constraints (2.5) and given that (x', z') is in the conditions of Proposition 1, we can derive:

$$\sum_{d \in D} z'^{d}(i, C) = \sum_{d \in D} z'^{d}(C, i) + \sum_{d \in D} z'^{d}_{di} - \sum_{d \in D} z'^{d}_{id} = \sum_{d \in D} z'^{d}(C, i) + x'(D, i) - x'(i, D).$$

Thus, by combining the two above equalities, we obtain:

$$\epsilon(i, C) = x'(i, C) - \sum_{d \in D} z'^{d}(i, C) = x'(C, i) - \sum_{d \in D} z'^{d}(C, i) = \epsilon(C, i).$$

Consider now an arbitrary depot $k \in D$ and a new solution (x', z'') where $z''^{d}_{ij} = z'^{d}_{ij}$ for all $d \in D \setminus \{k\}$ and for all $(i, j) \in A^C \cup A^d_O \cup A^d_I$, where $z''^{k}_{ij} = z'^{k}_{ij} + \epsilon_{ij}$ for all $(i, j) \in A^C$, and where $z''^{k}_{ij} = z'^{k}_{ij}$ for all $(i, j) \in A^d_O \cup A^d_I$. Observe that (x', z') and (x', z'') have the same cost given an objective function which only depends on the x variables. In order to complete the proof, we only need to show that (x', z'') is a feasible solution of the linear programming relaxation of model B.

Clearly, (x', z'') satisfies the assignment relaxation constraints (1.2)–(1.5) by definition. Additionally, and also by definition, (x', z'') satisfies constraints (2.3)–(2.4), the linking constraints (2.11) and the flow conservation constraints (2.5) for all $d \in D \setminus \{k\}$. Observe also that, by the definition of ϵ, the solution (x', z'') satisfies the linking constraints (2.10). Finally, in order to complete the proof, we now need to verify that (x', z'') satisfies the flow conservation constraints (2.5) for depot k.

Consider $i \in C$ and observe that

$$z''^{k}(k \cup C, i) = z'^{k}(k \cup C, i) + \epsilon(C, i) = z'^{k}(i, k \cup C) + \epsilon(i, C) = z''^{k}(i, k \cup C),$$

which follows from the fact that z' satisfies the flow conservation constraints (2.5) for depot k and from the result proved above for ϵ. This completes the proof since we have shown that (x', z'') is a solution of the linear programming relaxation of model B. □

The advantage of using the 3I^{++} system over the 3I^{+} system is that if we consider the equality version of the linking constraints, namely constraints (2.10)–(2.11), we can eliminate the x variables from the model and obtain a valid formulation for the multi-depot routing problem involving only the z variables if we re-define them as binary variables by using constraints (2.7), as in the four previous works by Albareda-Sambola et al. (2005), Bektaş (2012), Fernández & Rodríguez-Pereira (2016) and Hill & Voß (2016). In addition, the 3I^{++} system may improve the linear programming relaxation values if other cost functions are used, more specifically if the cost functions are functions of the z variables, since Proposition 2 requires that the cost function used only depends on the x variables.

We conclude by observing that neither the 3I^{+} system nor the 3I^{++} system is practical to use to solve large instances of the multi-depot routing problem, even if they contain substantially

less linking constraints than the 3I system. This is why the network flow interpretation, given in Section 2.3.2 for the original 3I system and above for the 3I$^+$ system and the 3I^{++} system, is important. We will see in Section 2.4 that, by using the max-flow/min-cut theorem, we will be able to prove an equivalence result between a set of exponentially-many multi-cut constraints in the space of the x variables and the 3I system. This result will also indirectly provide a polynomial-time separation algorithm for the multi-cut constraints and, additionally, we will be able to derive from the 3I$^+$ system a generalization of the multi-cut constraints.

2.3.4 A property of the systems of inequalities based on the z variables for symmetric cost instances

In this section we present a property of the 3I^{++} system that is related to the value of the linear programming relaxation of this system of inequalities for symmetric cost functions.

Proposition 3. *Suppose that c is a symmetric cost function and consider a solution x^* that satisfies the depot degree constraints* (1.2)–(1.3)*. Then, there exists a solution (x', z') that satisfies* (1.2)–(1.3) *and all the constraints of the 3I^{++} system and has the same cost as x^*.*

Proof. Let x' be defined as follows:

$$x'_{ij} = \frac{1}{2}x^*_{ij} + \frac{1}{2}x^*_{ji} \qquad \forall (i,j) \in A. \tag{2.12}$$

Clearly both x^* and x' have the same cost given that the costs are symmetric. Additionally, we can easily notice that, for all $d \in D$, we have $x'(d, C) = \frac{1}{2}x^*(d, C) + \frac{1}{2}x^*(C, d) = \frac{1}{2} + \frac{1}{2} = 1$, which means that x' satisfies the depot outdegree constraints (1.2). By using a similar reasoning we can also prove that x' satisfies the depot indegree constraints (1.3).

We will now prove that a flow z' exists such that (x', z') satisfies the several constraints of the 3I^{++} system. Observe that for these proofs we will use the linking constraints between the z and the x variables (2.10)–(2.11) of the 3I^{++} system.

Regarding the depot outdegree constraints (2.3) for the z variables, consider $d \in D$ and notice that $z'^d(d, C) = x'(d, C) = 1$, since we already proved that x' satisfies the depot outdegree constraints (1.2). Equivalently, we can use the same reasoning for the depot indegree constraints (2.4) for the z variables, given that x' satisfies the depot indegree constraints (1.3).

With respect to the flow conservation constraints (2.5), suppose that, for some $d \in D$ and $i \in C$, we have $z'^d(d \cup C, i) < z'^d(i, d \cup C)$. By adding $\sum_{k \in D: k \neq d} z'^k(k \cup C, i)$ to both sides of this inequality we obtain:

$$\sum_{k \in D: k \neq d} z'^k(k \cup C, i) + z'^d(d \cup C, i) < z'^d(i, d \cup C) + \sum_{k \in D: k \neq d} z'^k(k \cup C, i).$$

By using the linking constraints (2.10)–(2.11) of the 3I^{++} system we can write the above inequality as

$$x(V,i) < z'^{d}(i, d \cup C) + \sum_{k \in D: k \neq d} z'^{k}(k \cup C, i) \leq x(i, d \cup C) + x(D \setminus \{d\} \cup C, i).$$

Finally, notice that from the definition of x' (2.12), we have $x'(i, d \cup C) = x'(d \cup C, i)$ and, thus, the above inequality can be written as $x(V,i) < x(V,i)$ which is a contradiction. Therefore, $z'^{d}(d \cup C, i) \geq z'^{d}(i, d \cup C)$. By applying a similar reasoning we can prove that $z'^{d}(d \cup C, i) \leq z'^{d}(i, d \cup C)$, which completes the proof. □

The implications of this result are mainly of practical concern. For symmetric cost instances, this result states that if we consider an incomplete model in the space of the x variables for the multi-depot routing problem which does not have any set of path elimination constraints then, by using the 3I^{++} system to model path elimination constraints, the linear programming relaxation value is the same as the one of the incomplete model. Observe that this result is interesting also because it applies to any set of path elimination constraints which is implied by the 3I^{++} system, as for example the 3I and the 3I^{+} systems and other sets of constraints which will be presented throughout this part of the dissertation.

2.4 A new set of path elimination constraints in the space of the x variables

In this section we present the multi-cut constraints, which are a newly developed set of path elimination constraints in the space of the x variables. The multi-cut constraints are one of the most important contributions of this dissertation.

We present their basic version in Section 2.4.1. Then, in Section 2.4.2 we show that these new constraints are related to the arc-depot assignment variable based systems of inequalities of Section 2.3, namely through the max-flow/min-cut theorem. This relationship will then allow us to derive a generalization of the multi-cut constraints in Section 2.4.3.

2.4.1 The multi-cut constraints

We start this section by presenting a new set of path elimination constraints defined in the space of the x variables.

Proposition 4. *The following multi-cut constraints are valid for the multi-depot routing problem and eliminate circuits with two or more depots:*

$$x(S', d) + x(S', S) + x(d, S) \geq 1 \qquad \forall d \in D,\ \forall S \subset C. \tag{2.13}$$

Proof. Consider a depot $d \in D$ and a client subset $S \subset C$. To see that these constraints are valid, suppose that $x(S', d)$, $x(S', S)$ and $x(d, S)$ are all 0. In any feasible solution of the multi-depot routing problem there must exist an arc outgoing d and another arc ingoing d. Therefore, since $x(d, S) = 0$ and $x(S', d) = 0$, then $x(d, S') = 1$ and $x(S, d) = 1$, respectively. But then, since $x(S', S) = 0$ there is no way of closing the circuit. To see why circuits with two or more depots are eliminated consider, without loss of generality, that a circuit contains two depots, d_1 and d_2 from D, and define S' as the set of client nodes which are in the path between depot d_1 and depot d_2. Thus, $x(S', d_1) = x(d_1, S) = x(S', S) = 0$ and so constraint (2.13) for depot d_1 is violated. □

We refer to constraints (2.13) as the 1-MCC inequalities. The above result shows that these constraints model the generic path elimination constraints (1.7), however, they are in exponential number and so they require a separation algorithm to be used effectively in practice. In Section 3.3.2 we will show that they can be separated in an exact way and in polynomial time by resorting to max-flow/min-cut computations in an adequate graph.

In order to derive the separation algorithm for the 1-MCC inequalities (2.13) we will see in the next section that these constraints are related to the 3I system presented in Section 2.3.1, for which the interpretation of the 3I system as a network flow model discussed in Section 2.3.2 will be very important.

2.4.2 Establishing a relationship between the multi-cut constraints and the systems of inequalities based on the z variables

In this section we show that the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) presented in the previous section are equivalent to the 3I system proposed in Section 2.3.1 in terms of their corresponding linear programming relaxation. Recall that in Section 2.3.2 we showed that the 3I system could be interpreted as a network flow system in a special 3-layered graph. In particular, we observed that the 3I system can be seen as guaranteeing that 1 unit of flow is sent from each depot to its copy in the 3-layered graph. Following this, we can relate the 1-MCC inequalities (2.13) to the 3I system as follows.

Proposition 5. *The projection of the linear programming relaxation of the 3I system, comprised of constraints* (2.3)–(2.7)*, onto the space of the x variables is given by the 1-MCC inequalities* (2.13) *and* $x_{ij} \geq 0,\ \forall (i, j) \in A$.

Proof. This follows from the max-flow/min-cut theorem. Given the interpretation of the 3I system in the 3-layered graph, we redefine variables z^d_{id} and x_{id} as $z^d_{i\overline{d}}$ and $x_{i\overline{d}}$, respectively, where $\overline{d}$ is the copy of node $d \in D$ in the 3-layered graph. The max-flow/min-cut theorem indicates that, for each depot d, 1 unit of flow is sent from d to its copy $\overline{d}$, with arc capacities

given by the values of the x variables, if and only if every cut separating d from $\overline{d}$ has capacity at least 1. This last constraint is $x(d \cup S', \overline{d} \cup S) \geq 1$ for any $S \subset C$. Redefining the variables $x_{i\overline{d}}$ into x_{id}, we obtain the multi-cut constraints (2.13). □

The importance of this result is two-fold. Firstly, it shows that in theory the 1-MCC inequalities (2.13) must provide the same linear programming relaxation value as the 3I system, and in particular the result of Proposition 3, which discussed the value of the linear programming relaxation for symmetric instances, also applies to the 1-MCC inequalities (2.13). Secondly, the fact that the proof of Proposition 5 is based on the max-flow/min-cut theorem indicates that we can devise a polynomial-time separation algorithm for the 1-MCC inequalities (2.13) based on max-flow/min-cut computations. This algorithm will be presented in Section 3.3.2.

In Section 2.3.3 we showed that the 3I system could be strengthened. More precisely, we defined the 3I$^+$ system which dominates the 3I system and which has fewer linking constraints between the z and the x variables as the 3I system. Given the result of Proposition 5, it would be interesting to see which inequalities we can derive in the space of the x variables by using the 3I$^+$ system. This will be topic of the next section.

2.4.3 A generalization of the multi-cut constraints

In this section we present a generalization of the new multi-cut constraints, namely the 1-MCC inequalities $x(S', d)+x(S', S)+x(d, S) \geq 1$ (2.13). In addition, we show that these generalized multi-cut constraints can be related to the 3I$^+$ system presented in Section 2.3.3, in a similar way as the 1-MCC inequalities (2.13) are related to the 3I system of Section 2.3.1.

Notice that the 1-MCC inequalities (2.13) are defined for each depot $d \in D$. A generalization of these constraints can be obtained by using depot subsets with more than one depot in the multi-cut. More precisely, consider the following constraints:

$$x(S', D') + x(S', S) + x(D', S) \geq |D'| \qquad \forall D' \subseteq D,\ \forall S \subset C. \tag{2.14}$$

We denote the constraints above for sets D', where $|D'| = k$, by k-MCC inequalities. Clearly, if $k = 1$ we obtain the 1-MCC inequalities (2.13) as a special case. The result that follows shows that the k-MCC inequalities (2.14) can be derived from the 3I$^+$ system while also providing a proof of their validity.

Proposition 6. *The projection of the linear programming relaxation of the 3I$^+$ system, comprised of constraints* (2.3)–(2.5)*,* (2.8)–(2.9) *and* (2.7)*, onto the space of the x variables is included in the polyhedron defined by the k-MCC inequalities* (2.14) *and $x_{ij} \geq 0,\ \forall (i, j) \in A$.*

Proof. The proof follows from the max-flow/min-cut theorem and also from the fact that we can view the 3I$^+$ system in the 3-layered graph, as we observed in Section 2.3.3. Variables z^d_{id} and

x_{id} are redefined as $z^d_{i\bar{d}}$ and $x_{i\bar{d}}$, respectively, as in the proof of Proposition 5. For each subset $D' \subseteq D$, the max-flow/min-cut theorem indicates that $|D'|$ units of flow are sent from D' to the subset of the copies of the depots in D', say $\overline{D'}$, if and only if every cut separating D' from $\overline{D'}$ has capacity at least $|D'|$. This last constraint is expressed as $x(D' \cup S', \overline{D'} \cup S) \geq |D'|$ for any $S \subset C$. Redefining the variables $x_{i\bar{d}}$ into x_{id}, we obtain the k-MCC inequalities (2.14). □

Note that in this case it is not obvious whether or not we have a strict inclusion in the above result, since the linking constraints $\sum_{d \in D} z^d_{ij} \leq x_{ij}$ (2.8) for arcs in A^C aggregate multiple flows simultaneously. Therefore, in theory, we can only ensure that the k-MCC inequalities (2.14) provide a linear programming relaxation value which is at most the one of the 3I$^+$ system. In addition, note that the result of Proposition 3, regarding the linear programming relaxation value for symmetric instances, also applies to the k-MCC inequalities.

The proof of Proposition 6 indicates that it is also possible to devise a separation algorithm for the k-MCC inequalities (2.14) based on max-flow/min-cut computations. We will present an exact separation algorithm in Section 3.3.3 which, however, is not a polynomial-time algorithm and, thus, the use of the k-MCC inequalities (2.14) is not as straightforward as the 1-MCC inequalities (2.13). Nevertheless, we will also present in Section 3.3.3 a heuristic separation algorithm for the k-MCC inequalities (2.14) which we will show to be very effective in practice.

Next we prove a result which has implications on the reduction of the computational effort needed to separate the k-MCC inequalities (2.14).

Proposition 7. *In the presence of the assignment constraints* (1.2)–(1.5) *and non-negativity constraints* $x_{ij} \geq 0,\ \forall (i,j) \in A$*, every* $|D|$*-MCC inequality* (2.14) *is redundant and for each subset* $S_1 \subset C$ *and each subset* $D_1 \subset D$*, with* $|D_1| = k$*, the corresponding k-MCC inequality* (2.14) *is equivalent to the* $(|D| - k)$*-MCC inequality* (2.14) *written for* $S_2 = C \setminus S_1$ *and for* $D_2 = D \setminus D_1$.

Proof. We start by showing that the assignment constraints (1.2)–(1.5) imply the following equality:

$$x(C \setminus S_1, D_1) + x(C \setminus S_1, S_1) + x(D_1, S_1) + |D \setminus D_1| = \\ = |D_1| + x(S_1, D \setminus D_1) + x(S_1, C \setminus S_1) + x(D \setminus D_1, C \setminus S_1)\ \forall D_1 \subseteq D,\ \forall S_1 \subset C. \quad (2.15)$$

To see why, let $S_2 = C \setminus S_1$ and $D_2 = D \setminus D_1$. The proof is based on four valid equalities that are obtained by adding the assignment constraints (1.2)–(1.5) for adequate subsets and then combining the resulting equalities.

(i) by adding the depot outdegree constraints (1.2) for $d \in D_1$ and partitioning C into S_1 and S_2, we obtain $x(D_1, S_1) + x(D_1, S_2) = |D_1|$;

(ii) similarly, by adding the depot indegree constraints (1.3) for $d \in D_2$ and partitioning C into S_1 and S_2, we obtain $x(S_2, D_2) + x(S_1, D_2) = |D_2|$;

(iii) by adding the client outdegree constraints (1.4) for $i \in S_2$ and partitioning V into D_1, D_2, S_1 and S_2, we obtain $x(S_2, D_1) + x(S_2, D_2) + x(S_2, S_1) + x(S_2) = |S_2|$;

(iv) similarly, by adding the client indegree constraints (1.5) for $i \in S_2$ and partitioning V into D_1, D_2, S_1 and S_2, we obtain $x(D_1, S_2) + x(D_2, S_2) + x(S_1, S_2) + x(S_2) = |S_2|$.

Observe that by combining the two equalities under (iii) and (iv) we obtain the equality $x(S_2, D_1) + x(S_2, D_2) + x(S_2, S_1) = x(D_1, S_2) + x(D_2, S_2) + x(S_1, S_2)$. In this last equality we now use the first two equalities under (i) and (ii) and replace $x(S_2, D_2)$ by $|D_2| - x(S_1, D_2)$ and $x(D_1, S_2)$ by $|D_1| - x(D_1, S_1)$ to obtain (2.15) for the sets D_1 and S_1.

We can now prove the two statements of the main result. The first statement is a direct consequence of the equality (2.15) written for $D_1 = D$ and the fact that $x(S_1, C \setminus S_1) \geq 0$ for any $S_1 \subset C$. From equality (2.15) we also obtain that $\forall D_1 \subset D,\ \forall S_1 \subset C$,

$$x(S_2, D_1) + x(S_2, S_1) + x(D_1, S_1) \geq |D_1|$$

if and only if

$$x(S_1, D_2) + x(S_1, S_2) + x(D_2, S_2) \geq |D_2|,$$

where $D_2 = D \setminus D_1$ and $S_2 = C \setminus S_1$. □

Proposition 7 indicates that, in practice, it suffices to separate only half of the k-MCC inequalities (2.14), as those that are not separated will be implied by the ones that are. For example, for an instance with six depots, we only need to search for violated 1-MCC, 2-MCC and half of the 3-MCC inequalities. This is relevant for any exact separation algorithm for these constraints, however, this reduction still does not change the complexity of the exact separation algorithm that we will present in Section 3.3.3.

Finally, we observe that by using the depot indegree constraints (1.3) and the client indegree constraints (1.5), the k-MCC inequalities (2.14) can be rewritten as follows:

$$x(D', S) + x(S) + x(S, D \setminus D') \leq |S| \qquad \forall D' \subset D,\ \forall S \subset C. \tag{2.16}$$

This second form of writing the k-MCC inequalities (2.14) is interesting in order to more easily compare them to other path elimination constraints, such as the ones which will be presented in the next section.

2.5 Path elimination constraints from the literature

The discussion about path elimination constraints in the literature regarding problems which involve multiple depots is not extensive, as we have mentioned. In most recent works (see, e.g., Belenguer et al. 2011, Benavent & Martínez-Sykora 2013, Sundar & Rathinam 2017), the path elimination constraints used are mostly adaptations (and generalizations) of the so-called chain-barring constraints proposed earlier by Laporte et al. (1986). In order to use the chain-barring constraints in this dissertation they need to be adapted, since previous studies describe formulations on undirected graphs whereas this study is based on directed graphs.

In Section 2.5.1 we present the adaptation of the chain-barring constraints for directed graphs that we will use and then, in Section 2.5.2, we compare them to the multi-cut constraints presented in Section 2.4.

2.5.1 An adaptation of the chain-barring constraints

In this section we show how to adapt the chain-barring constraints, originally proposed by Laporte et al. (1986) for a location-routing problem, to the context of the multi-depot routing problem and, in particular, to models based on directed graphs. For that we will be using the path elimination constraints of Benavent & Martínez-Sykora (2013) as a reference point.

Previous studies model path elimination constraints in undirected graphs by using binary variables which indicate whether an edge of the graph is used or not. In particular, Benavent & Martínez-Sykora (2013) use a variable u_{ij}, defined as binary for every pair $i, j \in C,\ i \neq j$ to indicate whether or not the edge $\{i, j\}$ between clients i and j is used in the solution, and defined as {0,1,2} for every pair (d, i), where $d \in D$ and $i \in C$, indicating, respectively, whether the edge linking depot d and client i is not used or if it is used once or twice, with the latter case forming a two-node cycle, that is, a cycle with a depot and a client. The edge variables used by Benavent & Martínez-Sykora (2013) can be related to the directed variables x_{ij} through the equalities $u_{ij} = x_{ij} + x_{ji}$ for every edge $\{i, j\}$. Observe that any model which satisfies the generic subtour elimination constraints (1.6) satisfies, in particular, the inequalities $x_{ij} + x_{ji} \leq 1$ for $i, j \in C,\ i \neq j$, and thus the inequality $u_{ij} \leq 1$ is satisfied. This relationship allows us to write the path elimination constraints described by Benavent & Martínez-Sykora (2013) by using the arc variables x as follows:

$$x(D', i) + x(i, D') + 2x(S) + x(j, D \setminus D') + x(D \setminus D', j) \leq 2|S| - 1$$
$$\forall D' \subset D,\ \forall S \subset C : 3 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j. \quad (2.17)$$

The inequalities above, under the relationship between the undirected variables u and the directed variables x, correspond to the directed version of the path elimination constraints used by

Benavent & Martínez-Sykora (2013). However, these only eliminate paths between two different depots as long as the path has at least three client nodes, and so they must be complemented with additional sets of constraints for the other cases. Before we discuss that, and for completeness, we provide a proof of their validity and of the fact that they are path elimination constraints for the case in which the unfeasible path has at least three client nodes, both of which follow from arguments similar to the ones given by Benavent & Martínez-Sykora (2013).

Proposition 8. *Constraints* (2.17) *are valid for the multi-depot routing problem and eliminate paths with at least three client nodes between two different depots.*

Proof. We start by showing that they are valid. Consider a client subset with at least three nodes $S \subset C$, two nodes $i, j \in S,\ i \neq j$ and $D' \subset D$. Recall that $x(S) \leq |S| - 1$ since no client-only circuits can exist.

First we prove that if $x(S) < |S| - 1$, then constraints (2.17) are redundant. Clearly, if $x(S) < |S| - 2$ the constraints are redundant. Consider then that $x(S) = |S| - 2$ and suppose that $x(D', i) = x(i, D') = x(j, D \setminus D') = x(D \setminus D', j) = 1$. Then, because of the client degree constraints (1.4)–(1.5), neither i nor j are connected to nodes of S, hence, we can derive $x(S) = x(S \setminus \{i, j\}) \leq |S| - 3$, which is a contradiction. This means that the inequality $x(D', i) + x(i, D') + x(j, D \setminus D') + x(D \setminus D', j) \leq 3$ must hold and, thus, constraints (2.17) are redundant. Therefore, the only case of interest is when $x(S) = |S| - 1$.

Suppose now that $x(S) = |S| - 1$. This means that there is a path in S which links all nodes of S, including i and j. Then, in order not to have any unfeasible paths between depots, either i is linked to a depot in D' or j is linked to a depot in $D \setminus D'$ and so constraints (2.17) which read

$$x(D', i) + x(i, D') + x(j, D \setminus D') + x(D \setminus D', j) \leq 1$$

are valid.

To see why paths with at least three client nodes between two different depots are eliminated consider, without loss of generality, a path with at least three client nodes between two depots $d_1 \in D'$ and $d_2 \in D \setminus D'$ from D, define S as the set of client nodes which are in the path, and suppose that $i \in S$ is linked to d_1 and $j \in S$ is linked to d_2. Thus, $x(S) = |S| - 1$ and $x(D', i) + x(i, D') + x(j, D \setminus D') + x(D \setminus D', j) = 2$ and so there is a constraint (2.17) for this set S such that $d_1 \in D'$ and $d_2 \in D \setminus D'$ which is violated. □

Constraints (2.17) are not valid when $|S| = 2$ since the right-hand side is equal to 3 and it is feasible that both i and j are part of a two-node circuit. That is, circuits of the type (d_1, i, j, d_2, p, q, d_1), in which d_1 and d_2 are depots and i, j, p and q are clients, need to be eliminated by considering a different set of constraints. To eliminate such circuits, we can use the following constraints, which are the directed version of similar constraints described by Benavent &

Martínez-Sykora (2013):

$$x(D', i) + x(i, D') + 3x_{ij} + 3x_{ji} + x(j, D \setminus D') + x(D \setminus D', j) \leq 4 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j. \tag{2.18}$$

Once again, for completeness, we prove their validity and the fact they eliminate paths with exactly two client nodes between different depots.

Proposition 9. *Constraints* (2.18) *are valid for the multi-depot routing problem and eliminate paths with exactly two client nodes between two different depots.*

Proof. The validity of these constraints follows from similar arguments to the ones used in the proof of Proposition 8. Consider $D' \subset D$ and $i, j \in C,\ i \neq j$. Note that $x_{ij} + x_{ji} \leq 1$ in any feasible solution and that constraints (2.18) are only of interest when $x_{ij} + x_{ji} = 1$. In this case, i and j are linked, hence, either i is linked to a depot in D' or j is linked to a depot in $D \setminus D'$ and so constraints (2.18), which read $x(D', i) + x(i, D') + x(j, D \setminus D') + x(D \setminus D', j) \leq 1$, are valid.

To see why paths with exactly two client nodes between two different depots are eliminated consider a path (d_1, i, j, d_2), where $d_1, d_2 \in D,\ d_1 \neq d_2$ and $i, j \in C$. Then, $x_{d_1 i} = x_{j d_2} = 1$ and $3x_{ij} = 3$, thus a constraint (2.18) for nodes i and j and a depot subset D' such that $d_1 \in D'$ and $d_2 \in D \setminus D'$ is violated. □

Both sets of constraints (2.17) and (2.18) are in exponential number, however, we will present a polynomial-time separation algorithm in Section 3.3.4 based on a separation algorithm proposed by Belenguer et al. (2011) which resorts to max-flow/min-cut computations. This algorithm is not an exact separation algorithm, as will be explained in Section 3.3.4, however, it is very effective in practice.

2.5.2 A comparison to the multi-cut constraints

Since the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) presented in Section 2.4.3 and the directed chain-barring constraints (2.17)–(2.18) are both path elimination constraints defined in the space of the x variables, it is interesting to compare one set to the other.

To establish a theoretical connection, we start by writing the k-MCC inequalities (2.14) in their second form (2.16), which we presented at the end of Section 2.4.3, and which we recall is as follows:

$$x(D', S) + x(S) + x(S, D \setminus D') \leq |S| \qquad \forall D' \subset D,\ \forall S \subset C. \tag{2.16}$$

Consider, now, the following weaker version of (2.16)

$$x(D', i) + x(S) + x(j, D \setminus D') \leq |S| \qquad \forall D' \subset D,\ \forall S \subset C,\ \forall i, j \in S,\ i \neq j, \tag{2.19}$$

and add up a constraint (2.19) written for sets D' and S with the same constraint written for sets $D \setminus D'$ and the same set S, where i and j are swapped. This operation results in the following constraint:

$$x(D', i) + x(i, D') + 2x(S) + x(j, D \setminus D') + x(D \setminus D', j) \leq 2|S| \quad \forall D' \subset D,\ \forall S \subset C,\ \forall i, j \in S,\ i \neq j. \quad (2.20)$$

Clearly these constraints are redundant since they are obtained by adding two valid inequalities. In fact, we can easily observe that, from an integer point of view, they do not even eliminate unfeasible paths between two different depots. However, if $|S| \geq 3$ we can reduce the right-hand side of constraint (2.20) to $2|S| - 1$ and obtain exactly the set of constraints (2.17) which are the directed chain-barring constraints for $|S| \geq 3$.

Observe that in the previous section we showed that the directed chain-barring constraints (2.17)–(2.18) eliminate paths between two different depots provided that the path has at least two client nodes. However, circuits of the type $\overline{\sigma} = (d_1, i_1, d_2, i_2, \ldots, d_{m-1}, i_{m-1}, d_m, i_m, d_1)$, where $d_1, \ldots, d_m \in D$ and $i_1, \ldots, i_m \in C$, for $m \geq 2$, that is, unfeasible circuits which alternate between a depot and a client, are not eliminated by these constraints. As has been shown by Laporte et al. (1986), circuits of the type $\overline{\sigma}$ are never optimal if we consider symmetric costs, since there always exists a feasible solution that is cheaper or at most as costly. Here we provide a similar proof for the case of directed graphs.

Proposition 10. *If c is a symmetric cost function, then there exists a set of circuits which cost at most the same as an unfeasible circuit $\overline{\sigma} = (d_1, i_1, d_2, i_2, \ldots, d_{m-1}, i_{m-1}, d_m, i_m, d_1)$, where $d_1, \ldots, d_m \in D$ and $i_1, \ldots, i_m \in C$, for $m \geq 2$.*

Proof. Consider a set of circuits σ_1 which is comprised of m single-client return trips of the form (d_j, i_j, d_j) for $j \in \{1, \ldots, m\}$. Consider another set of circuits σ_2, also comprised of m return trips, but now in the form (d_j, i_{j-1}, d_j) for $j \in \{2, \ldots, m\}$ and (d_1, i_m, d_1). We claim that the cheapest of these two sets of circuits σ_1 and σ_2 is cheaper than the unfeasible circuit $\overline{\sigma}$. Let $\overline{c}$ be the cost of circuit $\overline{\sigma}$, and $c_1,\ c_2$ be the costs of the sets of circuits σ_1 and σ_2, respectively. Since costs are symmetric, then $2\overline{c} = c_1 + c_2$. Suppose, without loss of generality, that $c_1 \leq c_2$. Then $2\overline{c} \geq 2c_1$, hence $\overline{c} \geq c_1$. □

Since in most of the literature that uses constraints similar to the chain-barring constraints it is assumed that costs are symmetric, these depot-client alternating circuits did not arise. For asymmetric instances the result of Proposition 10 does not hold, hence, we need to use another set of constraints to eliminate these unfeasible circuits. This is in fact guaranteed by the 1-MCC inequalities (2.13) for $|S| = 1$, which, for clarity, we write as:

$$x_{di} + x(i, D \setminus d) \leq 1 \qquad \forall d \in D,\ \forall i \in C. \quad (2.21)$$

We conclude this section by observing that we will be performing some computational tests to compare the multi-cut constraints to the directed chain-barring constraints. In particular, we will see that the linear programming relaxation values are significantly different depending on whether the instance is a symmetric cost instance or an asymmetric cost instance. In fact, the result of Proposition 3, which we recall stated a property of the arc-depot assignment variable based systems of inequalities of Section 2.3 for symmetric instances, does not hold for the directed chain-barring constraints. However, for asymmetric instances the multi-cut constraints provide considerably higher linear programming relaxation values.

2.6 Problem variants

In this section we show how to adapt the generalized multi-cut constraints, namely the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14), to variants of multi-depot routing problems which may be of interest in other studies and that require path elimination constraints.

In the multi-depot routing problem all depots are necessarily used, however, a case of interest could be the case in which D is considered a set of potential depot locations and additional decisions have to be taken as to whether each depot $d \in D$ should be used or not. This variant is a simplification of the well-known location-routing problem (see, e.g., Laporte et al. 1986, 1988, Belenguer et al. 2011, Albareda-Sambola 2015), where there is often a fixed cost of using a depot and, for that reason, choosing which depots to use is an important decision.

Observe that in this problem variant a depot does not necessarily need to have any incident arcs. Therefore, we need to modify the depot degree constraints (1.2)–(1.3) to, respectively,

$$\sum_{j \in C} x_{dj} = y_d \qquad \forall d \in D \tag{2.22}$$

$$\sum_{j \in C} x_{jd} = y_d \qquad \forall d \in D, \tag{2.23}$$

where y is an additional binary variable such that $y_d = 1$ if depot $d \in D$ is used, and $y_d = 0$ otherwise. These new depot degree constraints state that the outdegree and the indegree of a depot $d \in D$ are 1 if d is used or, equivalently, if $y_d = 1$, whereas if d is not used, that is, if $y_d = 0$, then the outdegree and the indegree of d must be 0.

Clearly, path elimination constraints are still required to prevent unfeasible paths between depots which are used. For this purpose we can use an adaptation of the k-MCC inequalities (2.14) as follows:

$$x(S', D') + x(S', S) + x(D', S) \geq y(D') \qquad \forall D' \subseteq D,\ \forall S \subset C. \tag{2.24}$$

An easy way to see that the adaptation of the k-MCC inequalities (2.24) are a valid set of path elimination constraints is by using the new depot indegree constraints (2.23) and the client

indegree constraints (1.5) and rewrite them as

$$x(D', S) + x(S) + x(S, D \setminus D') \leq |S| \qquad \forall D' \subset D,\ \forall S \subset C, \tag{2.16}$$

which are the alternative form of the original k-MCC inequalities (2.14) presented in Section 2.4.3 and which are mathematically expressed in the same way in this problem variant.

The cut form of the adapted k-MCC inequalities (2.24) shows that the separation algorithm for these constraints is essentially the same as the one which we will present in Section 3.3.3 for the original k-MCC inequalities (2.14). In fact, we can prove similar results to the ones of Propositions 5 and 6, which we recall relate the original k-MCC inequalities (2.14) to arc-depot assignment variable based systems of inequalities. The proof of these similar results only differs in the fact that, in the 3-layered graph, the flow sent from a given depot $d \in D$ to its copy equals y_d instead of 1.

Another case which may be of interest in other studies is the case in which there are multiple vehicles at a given depot, for example due to vehicles having limited capacities. The adaptation in this case is simple given the previous discussion. More precisely, by changing the y variables from binary variables to general integer variables $y_d \in \{0, \ldots, k_d\}$, where k_d is the number of vehicles available at depot $d \in D$, we can simply use the depot degree constraints (2.22)–(2.23) and the adaptation of the k-MCC inequalities (2.24) above.

2.7 Concluding remarks

In this chapter we proposed a new formulation in the space of the arc variables x for the multi-depot routing problem. More precisely, we used the generic formulation of Chapter 1 defined by the assignment constraints (1.2)–(1.5) and the domain constraints for the x variables (1.8) to which we added a set of subtour elimination constraints and a set of path elimination constraints. The subtour elimination constraints are a straightforward adaptation of constraints described in the literature, whereas the path elimination constraints, namely the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13), are a newly developed set of constraints and are one of the most important contributions of this dissertation.

We also presented a network flow formulation, based on earlier formulations described in the literature, which is defined on a 3-layered graph. This 3-layered graph is obtained from the original graph by introducing a copy of each depot and replacing the arcs from the clients to the original depots by arcs from the clients to their copies. The network flow formulation ensures that, in the 3-layered graph, one unit of flow is sent from each depot to its copy, thus guaranteeing that unfeasible paths between two depots do not exist. In this chapter we were able to prove that the 1-MCC inequalities (2.13) correspond to the projection onto to the space of the x variables of this network flow formulation. The importance of this result is two-fold. Firstly,

the linear programming relaxation value of the network flow formulation is equal to the linear programming relaxation value of the formulation defined by the 1-MCC inequalities (2.13) (and non-negativity constraints) and, secondly, given that the proof of this relationship is based on the max-flow/min-cut theorem, we will be able to devise a polynomial-time exact separation algorithm for the new path elimination constraints.

Additionally, we strengthened the network flow formulation by using arguments based on the disjointness of the circuits for which the projection onto the space of the x variables resulted in a generalization of the 1-MCC inequalities (2.13). We also theoretically compared the new multi-cut constraints to a set of path elimination constraints which are used in the majority of the recent literature for related problems.

In Chapter 3 we will present a branch-and-cut algorithm based on the new formulation described in this chapter, which will allow us to assess the effectiveness of the new multi-cut constraints in practice.

Chapter 3

A branch-and-cut algorithm

Contents

3.1 Introduction

In this chapter we present a branch-and-cut algorithm for the multi-depot routing problem, based on the sets of constraints in the space of the x variables presented in Chapter 2, that we will computationally evaluate by using a number of test instances. The branch-and-cut algorithm (Padberg & Rinaldi 1987, 1991) is an important combinatorial optimization tool used to solve integer linear programming problems, which combines the branch-and-bound algorithm with cutting plane algorithms (see, e.g., Wolsey 1998, for an overview on the branch-and-bound and the cutting plane algorithms).

The main idea of a regular branch-and-bound algorithm is as follows. Suppose that we want to solve an integer linear programming problem in which we are minimizing a given objective function. We start by solving the linear programming relaxation of a formulation for the integer linear programming problem. If the solution of the linear programming relaxation is integer, then it is the optimal solution of the integer linear programming problem and the algorithm stops. Otherwise, we create a number of new problems derived from the original integer linear programming problem by partitioning the solution space. For example, the most usual way of doing this is by choosing a variable u with a fractional value u^* and partitioning the solution space in order to create two new problems such that in one we have $u \leq \lfloor u^* \rfloor$ and in the other we have $u \geq \lceil u^* \rceil$. Each of these problems is then treated as a new integer linear programming problem and the process is repeated. Each problem is called a node of the branch-and-bound tree, with the original problem being the root node. In order for the branch-and-bound algorithm to work adequately, we also need to use bounding techniques. Observe that any integer solution found at a node of the branch-and-bound tree is a feasible solution of the original problem (it is optimal if it is found at the root node), and its value is an upper bound for the optimal value. It is imperative that we keep track of the best solution found so far, which is called the incumbent. Since our goal is to find the optimal solution to the integer linear programming problem, we know that we only need to look for integer solutions which have a lower value than our current incumbent. If at a given node the linear programming relaxation value is above the value of the incumbent, then we are sure that any potential new node that would be generated from that node could never lead to an integer solution with a lower value than the current incumbent and, thus, we can prune the node.

There are two main ways of improving the performance of a branch-and-bound algorithm,

which is usually measured in terms of time taken to find the optimal solution and is intrinsically associated with the number of branch-and-bound nodes solved. Firstly, it is very important to produce good upper bounds, so that we can eliminate many potential subproblems that we would otherwise have to solve. While solving the several nodes of the branch-and-bound tree, the algorithm naturally produces upper bounds by finding new incumbents, however, we can also find feasible solutions and, consequently, potentially new upper bounds, by resorting to heuristic algorithms. Secondly, it is also extremely important that the formulation we use for the integer linear programming problem produces linear programming relaxation values which are as close to the optimal value as possible. This will allow the linear programming relaxation values at each node of the branch-and-bound tree to be as high as possible and, thus, to more likely be above the value of the incumbent and, consequently, for the node to be pruned. The branch-and-cut algorithm operates on the latter. Essentially, we improve the linear programming relaxation values at each node by using a cutting plane approach, that is, we generate valid inequalities for the integer linear programming problem which are violated by the optimal solution of the linear programming relaxation problem and then re-solve the linear programming relaxation problem to obtain an improved lower bound. This process is repeated until no more violated inequalities are found.

An important variant of the branch-and-cut algorithm occurs when the formulation we provide for the integer linear programming problem includes certain constraints which we do not wish to add at the start of the process. One of the reasons for this may be because the number of constraints is exponential in number. An important difference is that, whenever an integer solution is found at a given node, we must ensure that it is actually feasible for the original integer linear programming problem by checking if there are any violated constraints from the set of constraints which we left out of the initial formulation. If a violated constraint is found, we add it to the formulation so that any node which is solved from there on will not find that unfeasible solution again. There is an important observation to be made regarding this variation of the branch-and-cut algorithm. Since the formulation we are using is incomplete, it is expected that the linear programming relaxation values it provides are too far from the optimal value and, thus, it is extremely important that in the cutting plane phase we look for violated inequalities from the sets which were left out of the formulation in addition to other sets of valid inequalities.

When a general framework for the branch-and-cut algorithm was first proposed by Padberg & Rinaldi (1991), the processing power of contemporary computers was exponentially lower than modern ones and dedicated software to solve optimization problems, the so-called solvers, were still in their early stages. Nowadays, many modern solvers already have an efficient branch-and-cut algorithm framework embedded which is constantly improved with state-of-the-art techniques and that easily allows users to tailor it to their needs. Thus, unless there exists a very strong reason to implement a branch-and-cut algorithm from scratch, it is usually

much preferable to take advantage of the existing solvers. The branch-and-cut algorithm we will present in this chapter uses the underlying branch-and-cut algorithm framework of the CPLEX solver version 12.6.1. by IBM (2014).

This chapter is organized in the following way. In Section 3.2 we discuss what a modern branch-and-cut algorithm implementation consists of using as an example the 12.6.1 version of the CPLEX solver. Then, we present separation algorithms for the constraints of Chapter 2 which will be used in the branch-and-cut algorithm in Section 3.3. Afterwards, we present the set of test instances of the multi-depot routing problem that we use in the computational tests as well as software and hardware details in Section 3.4. In Section 3.5 we present some preliminary computational tests to help design the branch-and-cut algorithm. Next, in Section 3.6 we present the branch-and-cut algorithm for the multi-depot routing problem. In Section 3.7 we present some computational results and assess the performance of the branch-and-cut algorithm. Finally, we summarize the results of Section 3.7 and make some concluding remarks for this chapter in Section 3.8.

3.2 A modern branch-and-cut algorithm

In this section we explain the basics of using the branch-and-cut algorithm framework of the CPLEX solver to implement a branch-and-cut algorithm. In particular, we focus on defining some terminology used in this context. We observe that most of the terminology used in the context of the CPLEX solver is the same as, or similar to, the terminology used in other integer linear programming solvers.

Most commercial solvers in general, and CPLEX in particular, are able to solve an integer linear programming problem just by requiring a valid formulation of the problem. In fact, CPLEX has an implementation of a branch-and-cut algorithm which uses the basic idea of a branch-and-bound with cutting plane algorithms as well as many other state-of-the-art techniques, some of which their users are unaware of. Some known techniques include general purpose heuristics to try to find feasible solutions and general purpose cuts to improve the linear programming relaxation values in the cutting plane phase.

Observe that the requirement of using a valid formulation is more limiting than it appears since, essentially, the formulation we provide must be a compact formulation. CPLEX, however, allows for more advanced techniques. In particular, users can tailor the branch-and-cut algorithm by using what are called callback functions, which allow the user to modify how CPLEX should proceed in certain steps of the branch-and-cut algorithm.

More formally, at specific steps of the branch-and-cut algorithm, CPLEX checks if there is a callback function implemented by the user. If there is, then CPLEX temporarily stops the branch-and-cut algorithm and calls the callback function, and the algorithm continues in the code which

is defined in the callback function. This code is user made with no interference from CPLEX and, typically, it consists of asking for information from CPLEX, such as the current fractional or integer solution, the number of nodes which have been explored so far, etc., and, by using this information, applying some routine and then returning information to the solver.

There are several callback functions which can be used with CPLEX, however, in this dissertation we only explain three of them, which are the ones we will be using, namely the heuristic callback, the lazy constraint callback and the user cut callback.

3.2.1 Heuristic callback

A heuristic callback is a callback function that CPLEX calls whenever (i) a new linear programming relaxation solution is determined at the root node; and (ii) the final linear programming relaxation solution is determined at any node other than the root node. In other words, in the root node a heuristic callback is called every time a new linear programming relaxation solution is determined, even if it resulted from a re-optimization following the addition of violated inequalities, whereas in the remaining nodes the heuristic callback is only called after it has been proved that the linear programming relaxation solution does not violate any more valid inequalities.

The reason for the term heuristic is that the particular steps in which the heuristic callback is called are the most appropriate ones to look for a heuristic solution which can hopefully improve the current incumbent. In fact, many heuristics typically implemented in a heuristic callback use the information of the linear programming relaxation solution values (e.g., by attempting to round them to the nearest integer value). Note that if the current linear programming relaxation solution is an integer solution, then the heuristic callback is still called since the user may want to try to improve it (e.g. by using local search).

3.2.2 Lazy constraint callback

A lazy constraint callback is essential to use whenever the formulation we provide to CPLEX is incomplete, that is, whenever we leave a set of constraints out of the initial formulation (e.g., because they are exponentially-many). A lazy constraint callback is called whenever an integer solution is found, namely because the linear programming relaxation solution is integer or a heuristic solution was found by CPLEX.

Since the initial formulation given to CPLEX is not valid due to the constraints which were left out, then an integer solution is not necessarily a feasible solution. Thus, CPLEX calls any lazy constraint callback that has been implemented and lets the user decide whether the solution is feasible or not. The user must use the information of the integer solution and check for feasibility. If the solution is not feasible, then the user has the chance to inform CPLEX that a new constraint should be added to the formulation.

Note that the code in any lazy constraint callback is the user's responsibility and it must ensure that integer solutions are correctly identified as feasible or unfeasible. Usually this is done by using a dedicated separation algorithm for the set of constraints left out of the model which identifies a violated inequality if one exists.

3.2.3 User cut callback

The third and final type of callback function we will discuss is the user cut callback. These callback functions are called in the cutting plane phase whenever a new linear programming relaxation solution is found, including whenever it was obtained by re-optimizing a linear program to which valid inequalities were added. When CPLEX calls a user cut callback function, the control is passed onto the user who can then choose to search for and to add valid inequalities to further improve the linear programming relaxation value.

The code in a user cut callback is usually directed at collecting information related to the linear programming relaxation solution and then, by using that information, applying separation algorithms to determine violated inequalities if any exist. Note that the code in these particular callback functions is called many times during the whole branch-and-cut algorithm, hence, it is important that it is efficient. Good implementations of user cut callbacks require the use of techniques such as controlling which and how many valid inequalities are added before giving control back to CPLEX and designing efficient separation algorithms.

3.3 Separation algorithms

A separation algorithm for a given set of constraints is an algorithm which checks whether a given point, either fractional or integer, satisfies all the constraints in the set and, if not, identifies one of the violated inequalities. Separation algorithms are essential in order to use exponentially-sized sets of constraints in practice, usually by resorting to a cutting plane approach.

At this point it is important to distinguish effectiveness, that is, the ability to find violated inequalities, and efficiency, that is, the running time, of a separation algorithm. A separation algorithm is said to be exact if, given a point, it can identify an inequality violated by that point or prove that none exists with certainty. A separation algorithm is said to be heuristic if violated inequalities exist but the algorithm may not be able to find them. An exact separation algorithm is the most effective type of algorithm, since we can be sure that when the algorithm stops no more violated inequalities exist, however, it may be very inefficient. In fact, exact separation algorithms may not be polynomial-time algorithms and, even if they are, they can still be time consuming. Conversely, heuristic separation algorithms are not as effective, however, they can be designed to be as efficient as necessary. In the case where the exact separation algorithm

is not a polynomial-time algorithm, heuristic separation algorithms may be the only reasonable way to separate a given set of constraints. In addition, even if the exact separation algorithm is a polynomial-time algorithm, heuristic separation algorithms can help to considerably lower the overall running time by being used prior to resorting to the exact separation algorithm.

Another important distinction to make is whether the point we are separating is a fractional point or an integer point. Separation algorithms for fractional points also work for integer points, however, the converse is obviously not true. When designing separation algorithms it is important to find a dedicated algorithm for integer points since, usually, it is always possible to find an exact separation algorithm which is very efficient, namely because it only requires analyzing the integer point and checking for feasibility. In addition, the underlying idea of a separation algorithm for integer points can usually be adapted for fractional points in order to design an efficient heuristic separation algorithm. Following the discussion in Section 3.2, when using an integer linear programming solver, any separation algorithm for integer points should be implemented as a lazy constraint callback and any separation algorithm for fractional points should be implemented as a user cut callback.

In this section we present separation algorithms, both exact and heuristic, for the sets of constraints of exponential size presented in Chapter 2. Most of the exact separation algorithms presented in this chapter can be viewed as max-flow/min-cut problems in adequate auxiliary graphs. The separation algorithms which resort to max-flow/min-cut computations, given that the max-flow/min-cut problem can be solved in polynomial time, will be polynomial-time separation algorithms as long as the number of max-flow/min-cut computations required to be performed is also polynomial.

We will use two different auxiliary graphs. The first auxiliary graph used in the exact separation algorithms is denoted by st-extended graph and is a graph $W_1 = (V_1, A_1)$ which is obtained from the original graph G in the following way:

- The set of nodes V_1 includes all nodes of V and two additional nodes s and t, that is, $V_1 = V \cup \{s, t\}$;
- The set of arcs A_1 includes all arcs of A except the arcs ingoing the depots, additional arcs linking node s to every depot $d \in D$ and additional arcs linking every client $i \in C$ to node t, that is, $A_1 = (A \setminus \{(i, d) : i \in C,\ d \in D\}) \cup \{(s, d) : d \in D\} \cup \{(i, t) : i \in C\}$.

The other auxiliary graph used in the separation algorithms is denoted by st-extended 3-layered graph and is a graph $W_2 = (V_2, A_2)$, similar to the 3-layered graph described in Section 2.3.2, built in the following way:

- The set of nodes V_2 is comprised of every node of V and a copy of each depot $d \in D$. For simplification we will denote by $\overline{d}$ the copy of a depot $d \in D$ and by $\overline{D}$ the set of depot copies. In addition, we also add two nodes s and t. Therefore, $V_2 = V \cup \overline{D} \cup \{s, t\}$;

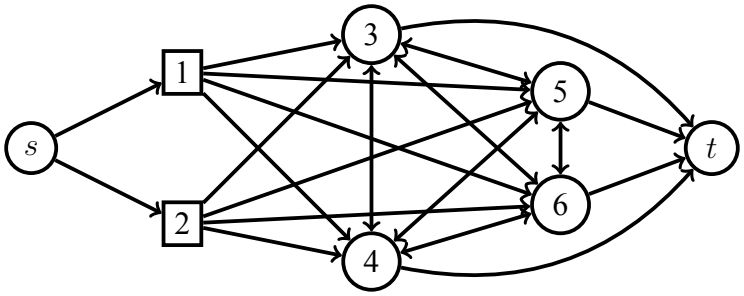

Figure 3.1: An example of the st-extended graph

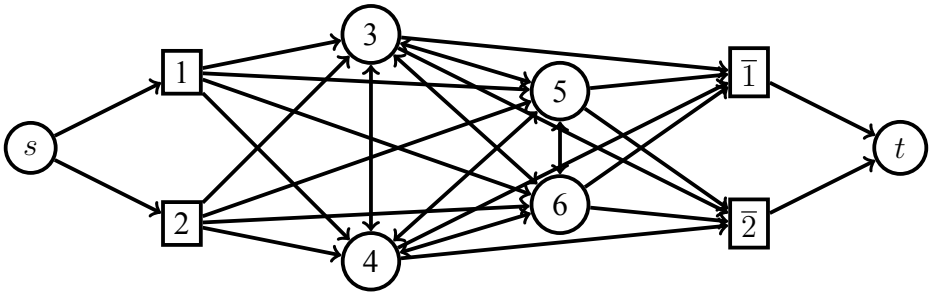

Figure 3.2: An example of the st-extended 3-layered graph

- The set of arcs A_2 is comprised of all arcs of A except the arcs ingoing the depots, additional arcs linking s to every depot $d \in D$, additional arcs linking every client $i \in C$ to every copy of a depot $\overline{d} \in \overline{D}$ and arcs linking every copy of a depot $\overline{d} \in \overline{D}$ to t, that is, $A_2 = (A \setminus \{(i,d) : i \in C,\ d \in D\}) \cup \{(s,d) : d \in D\} \cup \{(i,\overline{d}) : i \in C, \overline{d} \in \overline{D}\} \cup \{(\overline{d},t) : \overline{d} \in \overline{D}\}$.

Figures 3.1 and 3.2 show the auxiliary graphs W_1 and W_2 for a multi-depot routing problem in which $D = \{1,2\}$ and $C = \{3,4,5,6\}$, respectively.

3.3.1 Separation of the subtour elimination constraints (2.2)

In order to separate the subtour elimination constraints $x(D \cup S', S) \geq 1$ (2.2) we provide two algorithms: algorithm 3.1, which is a polynomial-time exact separation algorithm, and algorithm 3.2, which is a polynomial-time exact separation algorithm for integer points and a heuristic separation algorithm for fractional points.

Intuitively, observe that we can find violated subtour elimination constraints (2.2) by minimizing their left-hand side, which is a cut. Thus, algorithm 3.1 is based on max-flow/min-cut computations in the st-extended graph, with capacities given by the point being separated, and it checks whether or not the max-flow value from the set of depots to each client node is 1. If for any client the max-flow is less than 1, then the value of the min-cut is also less than 1 and a violated inequality is found.

As for algorithm 3.2 (see, e.g., Fischetti, Salazar-González & Toth 1997), the underlying

Algorithm 3.1

Require: A point x^* and the auxiliary st-extended graph.

1: **for all** $i \in C$ **do**
2: Set the capacities of the arcs $\{(s,d) : d \in D\}$ to 1, the capacities of the arcs $(p,q) \in A \setminus \{(k,d) : k \in C,\ d \in D\}$ to x^*_{pq}, the capacity of the arc (i,t) to 1 and the capacities of the arcs $\{(k,t) : k \in C,\ k \neq i\}$ to 0, and determine the maximum flow w from s to t.
3: **if** $w < 1$ **then**
4: The corresponding minimum cut defines a violated inequality (2.2) in which S is the subset of client nodes in the same shore as node t.
5: **end if**
6: **end for**

Algorithm 3.2

Require: A point x^*.

1: Find the connected components induced by x^* by considering all arcs $(p,q) \in A$ such that $x^*_{pq} > 0$ (e.g., by using a depth-first search algorithm).
2: **for all** connected components without any depot **do**
3: The set S comprised of the client nodes of the component defines a violated inequality (2.2).
4: **end for**

idea is to determine the connected components induced by the non-zero arc variables x and to check whether or not any of those components only includes client nodes. Note that any connected component of this form leads to an unfeasible subtour, regardless of whether the point being separated is fractional or integer. This algorithm is exact for integer points, but it is only heuristic for fractional points. Nevertheless, if a violated subtour elimination constraint (2.2) is found for a fractional point, then it is a maximally violated one since the left-hand side equals 0 and the right-hand side is 1, and, therefore, it is a very effective inequality to add to the model.

3.3.2 Separation of the 1-MCC inequalities (2.13)

In order to separate the 1-MCC inequalities $x(S',d) + x(S',S) + x(d,S) \geq 1$ (2.13) we provide two algorithms: algorithm 3.3, which is a polynomial-time exact separation algorithm, and algorithm 3.4, which is a polynomial-time exact separation algorithm for integer points and a heuristic separation algorithm for fractional points.

Intuitively, observe that we can find violated 1-MCC inequalities (2.13) by minimizing their left-hand side, which, as we showed in Section 2.4.2, is actually a cut in the 3-layered graph given the interpretation as a network flow model of the 3I system of Section 2.3.1. Thus, algorithm 3.3 is based on max-flow/min-cut computations in the st-extended 3-layered graph, with capacities given by the point being separated, and it checks whether or not the max-flow value from a

Algorithm 3.3

Require: A point x^* and the auxiliary st-extended 3-layered graph.

1: **for all** $d \in D$ **do**
2: Set the capacity of the arc (s, d) to 1, the capacities of the arcs $\{(s, d') : d' \neq d\}$ to 0, the capacities of the arcs $(p, q) \in A \setminus \{(k, d') : k \in C,\ d' \in D\}$ to x^*_{pq}, the capacities of the arcs $\{(k, \overline{d'}) : k \in C,\ \overline{d'} \in \overline{D}\}$ to $x^*_{kd'}$, the capacity of the arc $(\overline{d}, t)$ to 1 and the capacities of the arcs $\{(\overline{d'}, t) : \overline{d'} \neq \overline{d}\}$ to 0, and determine the maximum flow w from s to t.
3: **if** $w < 1$ **then**
4: The corresponding minimum cut defines a violated inequality (2.13) for d in which S is the subset of client nodes in the same shore as node t.
5: **end if**
6: **end for**

Algorithm 3.4

Require: A point x^*.

1: Find the connected components induced by x^* by considering all arcs $(p, q) \in A$ such that $x^*_{pq} > 0$ (e.g., by using a depth-first search algorithm).
2: **for all** connected components with two or more depots **do**
3: Find a path between two depots in the connected component, say a path from d_1 to d_2, and set S as the complementary set in C of the set of client nodes in the path found.
4: **if** x^* is integer **then**
5: S defines a violated inequality (2.13) for depot d_1.
6: **else**
7: If $x^*(d_1, S) + x^*(S) + x^*(S, D \setminus \{d_1\}) > |S|$, then S defines a violated inequality (2.13) for depot d_1.
8: **end if**
9: **end for**

depot $d \in D$ to its copy $\overline{d}$ is 1. If for any depot this max-flow value is below 1, then a violated inequality exists.

As for algorithm 3.4, the underlying idea is to first determine the connected components induced by the non-zero arc variables x and to check whether or not any of those components includes two or more depots. Then, for each of those components we look for an unfeasible path between two depots, which we do in the following way. If the point we are separating is an integer point, then finding such a path can be done by a depth-first search, however, this is not true for fractional points since one might be stuck in an infinite loop due to potential fractional circuits. For fractional points we apply a modified depth-first search in which nodes which have already been visited are removed from future potential nodes to search. This means that we may not be able to find an unfeasible path for a given connected component. Nevertheless, the computational tests show that this modified depth-first search is able to effectively find unfeasible

Algorithm 3.5

Require: A point x^* and the auxiliary st-extended 3-layered graph.

1: **for all** $D' \subset D$ such that $|D'| \leq \frac{|D|}{2}$ **do**
2: Set the capacities of the arcs $\{(s,d) : d \in D'\}$ to 1, the capacities of the arcs $\{(s,d) : d \in D \setminus D'\}$ to 0, the capacities of the arcs $(p,q) \in A \setminus \{(k,d) : k \in C,\ d \in D\}$ to x^*_{pq}, the capacities of the arcs $\{(k,\overline{d}) : k \in C,\ \overline{d} \in \overline{D}\}$ to x^*_{kd}, the capacities of the arcs $\{(\overline{d},t) : \overline{d} \in \overline{D'}\}$ to 1 and the capacities of the arcs $\{(\overline{d},t) : \overline{d} \in \overline{D} \setminus \overline{D'}\}$ to 0, and determine the maximum flow w from s to t.
3: **if** $w < |D'|$ **then**
4: The corresponding minimum cut defines a violated inequality (2.14) for D' in which S is the subset of client nodes in the same shore as node t.
5: **end if**
6: **end for**

Algorithm 3.6

Require: A point x^*.

1: Find a 1-MCC inequality (2.13) for a given set $S \subset C$ and a depot $d \in D$ which is violated by point x^* (e.g., by using algorithm 3.3 or 3.4).
2: **for all** $d' \in D \setminus \{d\}$ **do**
3: If $x^*(S',d') + x^*(d',S) < 1$, then add depot d' to the multi-cut.
4: **end for**

paths and, therefore, algorithm 3.4 is able to effectively find violated 1-MCC inequalities (2.13) for fractional points.

3.3.3 Separation of the k-MCC inequalities (2.14)

Algorithm 3.5 is an exact separation algorithm for the more general k-MCC inequalities $x(S',D') + x(S',S) + x(D',S) \geq |D'|$ (2.14) and it is very similar to the exact separation algorithm of the 1-MCC inequalities $x(S',d) + x(S',S) + x(d,S) \geq 1$ (2.13) described in algorithm 3.3. The only difference is that algorithm 3.5 is now based on Proposition 6 presented in Section 2.4.3, which relates the k-MCC inequalities (2.14) to the network flow interpretation of the 3I$^+$ system of Section 2.3.3. However, unless we fix the value of k, this exact separation algorithm is not a polynomial-time algorithm since a max-flow/min-cut computation has to be performed for many of the subsets $D' \subset D$, about half of them given the result of Proposition 7. Nevertheless, observe that the 1-MCC inequalities (2.13) suffice as path elimination constraints, and can be separated in polynomial-time, hence, the separation of the k-MCC inequalities (2.14) for $k \geq 2$ is optional.

Since algorithm 3.5 is not polynomial in time, we will not be using it in the branch-and-cut algorithm. However we did implement it and, in fact, computational testing shows that it is very inefficient. Instead, we will use the heuristic separation algorithm described in algorithm 3.6,

which we will show, based on some computational tests, to be very effective at finding violated k-MCC inequalities (2.14) for $k \geq 2$. Intuitively, suppose a violated 1-MCC inequality (2.13) for a depot $d \in D$ was found, for instance by using algorithm 3.3 or algorithm 3.4. Then, we can check for each depot $d' \neq d$ whether or not the violation of the multi-cut would increase by adding the depot d' to the multi-cut.

Observe that, even though algorithm 3.6 is a heuristic separation algorithm, once we determine the violated 1-MCC inequality (2.13) in step 1 and, consequently, the set S, this algorithm becomes exact for that specific set S.

3.3.4 Separation of the directed chain-barring constraints (2.17)–(2.18)

In order to separate the directed chain-barring constraints (2.17)–(2.18) presented in Section 2.5.1, which we recall are as follows

$$x(D', i) + x(i, D') + 2x(S) + x(j, D \setminus D') + x(D \setminus D', j) \leq 2|S| - 1$$
$$\forall D' \subset D,\ \forall S \subset C : 3 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j \quad (2.17)$$

$$x(D', i) + x(i, D') + 3x_{ij} + 3x_{ji} + x(j, D \setminus D') + x(D \setminus D', j) \leq 4$$
$$\forall D' \subset D,\ \forall i, j \in C,\ i \neq j, \quad (2.18)$$

we will resort to two different algorithms, one for fractional points and another for integer points which may also be used as a heuristic separation algorithm for fractional points.

First notice that, by using the client indegree constraints (1.5), constraints (2.17) can be rewritten as follows:

$$2x(D \cup S', S) \geq x(D', i) + x(i, D') + x(j, D \setminus D') + x(j, D \setminus D') + 1$$
$$\forall D' \subset D,\ \forall S \subset C : 3 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j. \quad (3.1)$$

This way of writing constraints (2.17), which was observed by Belenguer et al. (2011) for a different adaptation of the chain-barring constraints, shows us that we can separate them by minimizing their left-hand side and maximizing their right-hand side independently, since the set S only appears on the left-hand side and the choice of the set D' is only important for the right-hand side. Based on this observation, we define algorithm 3.7 for fractional points. Intuitively, we minimize the left-hand side by resorting to max-flow/min-cut computations in the st-extended graph, since it is a cut, with capacities given by the point being separated, and we maximize the right-hand side by inspection, that is, by checking for each depot $d \in D$ whether we should have $d \in D'$ or $d \in D \setminus D'$.

Algorithm 3.7 is an adaptation of an algorithm proposed by Belenguer et al. (2011) and it is a very effective separation algorithm for the directed chain-barring constraints (2.17)–(2.18), however, it is not exact. More precisely, the way in which the set D' is determined is exact, but

Algorithm 3.7

Require: A point x^* and the auxiliary st-extended graph.

1: **for all** $i, j \in C,\ i \neq j$ **do**
2: Set the capacities of the arcs $\{(s,d) : d \in D\}$ to 1, the capacities of the arcs $(p,q) \in A \setminus \{(k,d) : k \in C,\ d \in D\}$ to x^*_{pq}, the capacities of the arcs (i,t) and (j,t) to 1 and the capacities of the arcs $\{(k,t) : k \in C,\ k \neq i,\ k \neq j\}$ to 0. Then, determine the maximum flow w^{lhs} from s to t and define S as the client nodes in the same shore as node t in the corresponding minimum cut.
3: Set $w^{rhs} = 0$ and $D' = \emptyset$.
4: **for all** $d \in D$ **do**
5: If $x^*_{di} + x^*_{id} > x^*_{dj} + x^*_{jd}$, then set $w^{rhs} = w^{rhs} + x^*_{di} + x^*_{id}$ and $D' = D' \cup \{d\}$. Otherwise, set $w^{rhs} = w^{rhs} + x^*_{dj} + x^*_{jd}$.
6: **end for**
7: **if** $w^{lhs} < w^{rhs}$ **then**
8: **if** $|S| = 2$ **then**
9: Check if constraint (2.18) for the sets $S = \{i, j\}$ and D' determined is violated.
10: **else**
11: S and D' define a violated inequality (2.17).
12: **end if**
13: **end if**
14: **end for**

the way in which the set S is determined is not. This is due to the fact that there are additional conditions on the set S, in that $|S| \geq 3$ for constraints (2.17) and $|S| = 2$ for constraints (2.18), which are not taken into account when computing the max-flow/min-cut. Note that an exact separation algorithm for the directed chain-barring constraints (2.17)–(2.18) exists, however, it is more time-consuming and the extra time taken does not compensate using it instead of the one just presented since, as we mentioned, the algorithm 3.7 is already very effective.

In order to separate integer points we use algorithm 3.8. We also use a slight adaptation of this algorithm, namely algorithm 3.9, as a heuristic separation algorithm for fractional points. Observe that algorithm 3.8 already takes into account the discussion regarding asymmetric and symmetric costs of Section 2.5.2, that is, if costs are asymmetric and $|S| = 1$, then it checks to see if a 1-MCC inequality $x_{di} + x(i, D \setminus d)$ (2.21) for $|S| = 1$ is violated by the given point. Note that, in both algorithms, step 3 is the same as step 3 of algorithm 3.4, which we solve by using the modified depth-first search which we described for algorithm 3.4.

Algorithm 3.8

Require: An integer point x^*.

1: Find the connected components induced by x^* by considering all arcs $(p, q) \in A$ such that $x^*_{pq} > 0$ (e.g., by using a depth-first search algorithm).
2: **for all** connected components with two or more depots **do**
3: Find a path between two depots in the connected component, say a path from d_1 to d_2, and set S as the set of client nodes in the path found, i as the node directly linked to d_1 and j as the node directly linked to d_2 (i and j can potentially be the same node).
4: **if** $|S| \geq 2$ **then**
5: S, i and j define a violated inequality (2.17) or (2.18), depending on $|S|$, for $D' = \{d_1\}$.
6: **else**
7: **if** costs are asymmetric **then**
8: $S = \{i\} = \{j\}$ defines a violated inequality (2.21) for $D' = \{d_1\}$.
9: **end if**
10: **end if**
11: **end for**

3.4 Test instances and software/hardware configurations

In what remains of this chapter we will be mostly reporting on some computational experiments. Four sets, A, B, C and T, of instances are used in the experiments. The first set A is a subset of the symmetric benchmark location-routing problem instances, also used by Benavent & Martínez-Sykora (2013) and available at `http://prodhonc.free.fr/`, comprising a total of nine instances, six with 100 clients and with either 5 or 10 depots, and three with 200 clients and 10 depots. In these instances, we are given coordinates for each node and, following Benavent & Martínez-Sykora (2013), the costs c_{ij} and c_{ji} are determined as the Euclidean distance between nodes i and j multiplied by 100 and rounded up to the nearest integer. These instances are named as $|C|$-$|D|$-t, where $t = 1, 2, 3$ indicates how the points are distributed, where 1 represents uniform distribution, and 2 and 3 denote two and three clusters, respectively.

The instances in set B are also symmetric, where the node coordinates have been generated by uniformly placing as many points as needed in a 200×200 continuous grid. The set includes a total of 33 instances, comprising nine with 100 clients, nine with 200 clients and fifteen with 300 clients. For the first subset, we initially generate an instance with 100 clients and 20 depots. Removing 10 depots from this instance gives rise to another instance with 10 depots. Finally, a further removal of five depots results in an instance with five depots. The construction is used to obtain a geographically interrelated set of instances with the same set of clients, and where the depot locations have a nested structure. By repeating this process twice more, we obtain a total of nine 100-client instances. The same process has been used to generate nine instances with 200 clients and either 10, 20 or 40 depots, and fifteen instances with 300 clients and either 10,

Algorithm 3.9

Require: A fractional point x^*.

1: Find the connected components induced by x^* by considering all arcs $(p, q) \in A$ such that $x^*_{pq} > 0$ (e.g., by using a depth-first search algorithm).
2: **for all** connected components with two or more depots **do**
3: Find a path between two depots in the connected component, say a path from d_1 to d_2, and set S as the set of client nodes in the path found, i as the node directly linked to d_1 and j as the node directly linked to d_2 (i and j can potentially be the same node).
4: **if** $|S| \geq 2$ **then**
5: **if** $|S| > 2$ **then**
6: If $x^*(d_1, i) + x^*(i, d_1) + 2x^*(S) + x^*(j, D \setminus \{d_1\}) + x^*(D \setminus \{d_1\}, j) > 2|S| - 1$, then S, i and j define a violated inequality (2.17) for $D' = \{d_1\}$.
7: **else**
8: If $x^*(d_1, i) + x^*(i, d_1) + 3x^*_{ij} + 3x^*_{ji} + x^*(j, D \setminus \{d_1\}) + x^*(D \setminus \{d_1\}, j) > 4$, then i and j define a violated inequality (2.18) for $D' = \{d_1\}$.
9: **end if**
10: **end if**
11: **end for**

20, 30, 40 or 60 depots. The instances are named as bgs-100-$|D|$-t, where $t = 1, 2, 3$ denotes the instance number.

The instances in set C are asymmetric and are obtained by transforming the symmetric instances in set B. The transformation consists in increasing or decreasing the cost of an arc (i, j) by a percentage $p_{ij} \in [0.25, 0.75]$. More precisely, for each edge $\{i, j\}$, we randomly choose either arc (i, j) or arc (j, i) with equal probability. The cost of the chosen arc is increased, and the cost of the arc in the opposite direction is decreased by the specified percentage. The instances in set C are named as the original symmetric instance from which they derive with an added suffix "a".

Finally, the set T of instances is comprised of small-sized test instances, both symmetric and asymmetric, with 50 clients and between 2 and 20 depots, for a total of 38 instances. The symmetric instances were generated in the same way as the instances of set B and the asymmetric instances were generated in the same way as the instances of set C. The instances in set T are named as t-bgs-50-$|D|$ and as t-bgs-50-$|D|$a for the symmetric and asymmetric instances respectively.

All the computational results obtained were conducted on a single thread of an Intel Core i7-4790 3.6GHz processor in a personal computer with 8GB of RAM and within which CPLEX 12.6.1 Concert Technology for C++ was used. All of the code is original, except for the max-flow algorithm which is based on the push-relabel algorithm by Goldberg & Tarjan (1988).

3.5 Preliminary computational experiment

In this section we report on some preliminary computational tests to help us design the branch-and-cut algorithm. The formulation used in the branch-and-cut algorithm has as the underlying set of subtour elimination constraints the subtour elimination constraints $x(D' \cup S', S) \geq 1$ (2.2). As for path elimination constraints, there were two different sets which were presented, namely the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) and the directed chain-barring constraints (2.17)–(2.18), with the addition of the 1-MCC inequalities for $|S| = 1$ in the case of asymmetric instances, presented in Section 2.5.1. For the branch-and-cut algorithm we will use the 1-MCC inequalities (2.13) and we briefly justify this choice in Section 3.5.1.

In Section 3.5.2 we show how the use of valid inequalities affects the solution times for obtaining the optimal solution of instances of the multi-depot routing problem. In particular, we test the use of the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) and of the directed chain-barring constraints (2.17)–(2.18), the latter which are not used as path elimination constraints in the branch-and-cut algorithm but can still be used as valid inequalities.

3.5.1 Comparing the directed chain-barring constraints to the multi-cut constraints

In this section we compare the linear programming relaxation values of two formulations defined in the space of the arc variables x which only differ in the set of path elimination constraints used. Consider the generic model presented in Section 1.3 in which the subtour elimination constraints (2.2) are used to model the generic subtour elimination constraints (1.6), that is, the following model:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1.1}$$

$$\text{subject to:} \quad \sum_{j\in C} x_{dj} = 1 \qquad \forall d \in D \tag{1.2}$$

$$\sum_{j\in C} x_{jd} = 1 \qquad \forall d \in D \tag{1.3}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in C \tag{1.4}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in C \tag{1.5}$$

$$x(D \cup S', S) \geq 1 \qquad \forall S \subset C:\ 2 \leq |S| \leq |C| - |D| \tag{2.2}$$

$$\{(i,j) \in A : x_{ij} = 1\} \text{ contains no circuit with two or more depots} \tag{1.7}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{1.8}$$

The first formulation that we will consider is the formulation in which in the model above we replace the generic path elimination constraints (1.7) by the newly developed 1-MCC inequalities $x(S',d) + x(S',S) + x(d,S) \geq 1$ (2.13), which we will denote by MC formulation. The second formulation is obtained by replacing the generic path elimination constraints (1.7) with the directed chain-barring constraints (2.17)–(2.18), with the addition of the 1-MCC inequalities for $|S| = 1$ in the case of asymmetric instances, and it will be denoted as CB formulation.

We will compare the linear programming relaxation values given by both formulations in the test instances of set T. The linear programming relaxation values reported in this section are the real linear programming relaxation values because we deactivate all of CPLEX's general purpose cuts and preprocessing. Note that for now we are not interested in running times and, thus, the addition of general purpose cuts by CPLEX would not inform us correctly on the value of the linear programming relaxations. In addition, we use only the exact separation algorithm 3.3 for the 1-MCC inequalities (2.13) and the separation algorithm 3.7 for the directed chain-barring constraints (2.17)–(2.18). Note also that the optimal values reported were obtained with the branch-and-cut algorithm that we will present further on.

Table 3.1 shows the linear programming relaxation values of the MC formulation and of the CB formulation for the asymmetric and the symmetric instances of the instance set T. Table 3.1 reports the results divided into two smaller and similar tables. The left half is with respect to the asymmetric instances and the right half to the symmetric ones. In each half the first column indicates the instance name and the second column its optimal value (OPT). The other two columns indicate the linear programming relaxation value of the MC formulation (MC) and of the CB formulation (CB), respectively.

The results reported on the left half of Table 3.1 for the asymmetric instances show that the MC formulation consistently provides higher linear programming relaxation values than the CB formulation, except for the instances with a number of depots between 2 and 7 where the linear programming relaxation values coincide. Conversely, from the results on the right half, which are with respect to the symmetric instances, we see that the roles are reversed, that is, the CB formulation consistently provides higher linear programming relaxation values than the MC formulation, except in the instances with a number of depots between 2 and 7.

Choosing between the MC and the CB formulations is not straightforward. In fact, they seem to be complementary since the former provides higher linear programming relaxation values for asymmetric instances and the latter for symmetric instances. However, there is one reason why we believe the MC formulation is more suited to be the formulation to use in our branch-and-cut algorithm which is related to the separation algorithms in use. The separation algorithm used to identify violated directed chain-barring constraints (2.17)–(2.18) described in algorithm 3.7 is more time-consuming than the separation algorithm described in algorithm 3.3 for finding

Table 3.1: Comparing the linear programming relaxation values of the MC and the CB formulations

Name	OPT	MC	CB	Name	OPT	MC	CB
t-bgs-50-20a	821	819.9	800.469	t-bgs-50-20	1362	1324	1331.83
t-bgs-50-19a	824	822.9	803.655	t-bgs-50-19	1367	1325	1332.42
t-bgs-50-18a	807	805.4	788.55	t-bgs-50-18	1342	1300	1306.51
t-bgs-50-17a	784	784	770.958	t-bgs-50-17	1311	1283.5	1289.81
t-bgs-50-16a	752	752	738.958	t-bgs-50-16	1263	1236.5	1242.65
t-bgs-50-15a	754	746	738.1	t-bgs-50-15	1251	1234.5	1238.83
t-bgs-50-14a	757	748	737.611	t-bgs-50-14	1241	1224.5	1228.83
t-bgs-50-13a	749	737.286	728	t-bgs-50-13	1229	1217.5	1221.33
t-bgs-50-12a	741	729.188	720	t-bgs-50-12	1232	1220.5	1224.33
t-bgs-50-11a	722	715.714	708.417	t-bgs-50-11	1229	1198	1204.6
t-bgs-50-10a	719	708.038	702.548	t-bgs-50-10	1188	1163	1166
t-bgs-50-9a	707	695.531	690.188	t-bgs-50-9	1193	1166	1168.67
t-bgs-50-8a	689	679	675	t-bgs-50-8	1170	1145.5	1150.88
t-bgs-50-7a	679	670	670	t-bgs-50-7	1159	1137.5	1137.63
t-bgs-50-6a	659	651.167	651.167	t-bgs-50-6	1128	1113	1113
t-bgs-50-5a	648	639.75	639.75	t-bgs-50-5	1105	1091.65	1091.65
t-bgs-50-4a	621	610.563	610.563	t-bgs-50-4	1053	1050.25	1050.25
t-bgs-50-3a	628	620.846	620.846	t-bgs-50-3	1077	1073.88	1073.88
t-bgs-50-2a	628	619.75	619.75	t-bgs-50-2	1078	1074.75	1074.75

violated 1-MCC inequalities (2.13). More precisely, if MF is the complexity of the max-flow algorithm, then separating the 1-MCC inequalities (2.13) has a worst-case computational complexity of $O(MF \times |D|)$, since one max-flow has to be calculated for each depot, whereas separating the directed chain-barring constraints (2.17)–(2.18) has a worst-case computational complexity of $O(MF \times |C|^2)$, since a max-flow has to be calculated for each pair of clients. This means that, in practice, we need to rely on heuristic separation algorithms for the directed chain-barring constraints (2.17)–(2.18) which are never as effective as exact ones (or near-exact in this specific case), whereas separating the 1-MCC inequalities at every branch-and-bound node in an exact way is viable given their efficient separation. Note that this was also observed by Benavent & Martínez-Sykora (2013) and, in fact, they use heuristic separation algorithms to separate the undirected variant of the directed chain-barring constraints (2.17)–(2.18) rather than exact ones. Nevertheless, the linear programming relaxation values of the CB formulation are higher than the ones of the MC formulation for symmetric instances and, for that reason, we will test in the next section if we can take advantage of that by using the directed chain-barring constraints (2.17)–(2.18) as valid inequalities.

3.5.2 Evaluating the effectiveness of using valid inequalities

As we mentioned, we will use the MC formulation defined in Section 3.5.1 as the underlying formulation for the branch-and-cut algorithm. However, it does not mean that the other inequalities in the space of the x variables that we presented in Chapter 2 are not useful. In this section, we show some results with the purpose of seeing whether or not the use of certain inequalities as valid inequalities decreases the optimal solution times of a branch-and-cut algorithm based on the MC formulation. More precisely, we consider three variants of the MC formulation, namely the MC formulation in which:

- the directed chain-barring constraints (2.17)–(2.18) are added as valid inequalities by using the separation algorithm 3.7, denoted by MC+CB;
- the directed chain-barring constraints (2.17)–(2.18) are added as valid inequalities by using the heuristic separation algorithm 3.9, denoted by MC+hCB;
- the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ are added as valid inequalities by using the heuristic separation algorithm 3.5 every time a violated 1-MCC inequality (2.13) is found, denoted by k-MC.

We consider two different schemes. In the first scheme we separate the additional valid inequalities in every node of the branch-and-bound tree, whereas in the second scheme we only separate the valid inequalities in the root node of the branch-and-bound tree. To obtain these results we use the default settings of the branch-and-cut algorithm framework of CPLEX and provide all the necessary separation algorithms as callback functions, including the heuristic separation algorithms for the subtour elimination constraints $x(D \cup S', S) \geq 1$ (2.2) and the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) present in the MC formulation.

Table 3.2 shows the comparison results and its format is as follows. It is divided into two smaller and similar tables such that in the top half we have the case in which the valid inequalities are separated in every node of the branch-and-bound tree, whereas in the bottom half only in the root node. The first column indicates the name of the instance. The following eight columns are divided into four two-column parts, which are with respect to each of the four formulations being compared, namely, in order, the MC formulation, the MC+CB formulation, the MC+hCB formulation and the k-MC formulation. Each two-column pair reports the optimal value or the best lower and upper bound at the end of the time limit of the instance (OPT) and the time taken (t), in seconds, for the branch-and-cut algorithm to end. We set a time limit of 1200 seconds and used a subset of the instances with 200 clients of sets B and C, which include both asymmetric and symmetric instances. For the MC+CB and the MC+hCB formulations, we only ran the tests for the symmetric instances given the computational results presented in the previous section.

Table 3.2: Comparing the solution times of the MC, the MC+CB, the MC+hCB and the k-MC formulations

	MC		MC+CB		MC+hCB		k-MC	
Name	OPT	t	OPT	t	OPT	t	OPT	t
bgs-200-10-1a	1325	27	-	-	-	-	1325	16
bgs-200-20-1a	1361	279	-	-	-	-	1361	277
bgs-200-40-1a	1504	637	-	-	-	-	1504	178
bgs-200-10-1	2301	489	[2292, —]	1200*	2301	152	2301	447
bgs-200-20-1	2332	475	[2311, —]	1200*	2332	303	2332	337
bgs-200-40-1	2417	223	[2402, —]	1200*	2417	305	2417	144
Name	OPT	t	OPT	t	OPT	t	OPT	t
bgs-200-10-1a	1325	27	-	-	-	-	1325	17
bgs-200-20-1a	1361	279	-	-	-	-	1361	354
bgs-200-40-1a	1504	637	-	-	-	-	1504	150
bgs-200-10-1	2301	489	2301	668	2301	150	2301	436
bgs-200-20-1	2332	475	[2324, 2333]	1200*	2332	298	2332	524
bgs-200-40-1	2417	223	2417	596	2417	172	2417	244

*Not solved to optimality within the time limit.

These results provide several interesting conclusions. Starting with the k-MC formulation, we can see that separating the k-MCC inequalities (2.14) for $k \geq 2$ by using the heuristic separation algorithm 3.6 proposed turned out to be effective. In fact, the computational times were reduced in all cases when compared to the regular MC formulation when we separate them in every node of the branch-and-bound tree. This reduction is more significant as the number of depots increase, for which a possible explanation could be that, since there are more depots, there are more violated path elimination constraints found. Observe that the reduction in computational time was not substantial in the sense that the solution times remained in the same order of magnitude, however, the heuristic separation algorithm 3.6 is computationally very fast and easy to implement, hence, it seems evident from these results that adding the k-MCC inequalities (2.14) for $k \geq 2$ as valid inequalities via the use of the proposed heuristic algorithm is certainly a valid option.

Regarding the MC+CB and the MC+hCB formulations, we have completely opposite results between the two. For the MC+CB formulation, the results show that it does not seem useful to separate the directed chain-barring constraints (2.17)–(2.18) by using the separation algorithm 3.7. This was expected since this separation algorithm is very time-consuming, as we already noted. However, the MC+hCB formulation has a much different behavior and, in fact, it was able to improve the solution times in all instances, when we separated the directed chain-barring constraints (2.17)–(2.18) in the root node, and in 2 out of 3 instances, when we separated them in

all nodes of the branch-and-bound tree. Therefore, using the directed chain-barring constraints (2.17)–(2.18) as valid inequalities for symmetric instances is a valid option, specially if the heuristic separation algorithm is used only in the root node.

3.6 The outline of the branch-and-cut algorithm

In this section we describe the branch-and-cut algorithm proposed to solve the multi-depot routing problem based on the branch-and-cut algorithm framework of the CPLEX solver.

We start by detailing the several parts that compose the branch-and-cut algorithm, starting in Section 3.6.1 with the formulation which serves as the underlying formulation for the branch-and-cut algorithm as well as the valid inequalities that we use to increase the linear programming relaxation values.

Then, we discuss some other practical concerns that one must have in order to design a modern branch-and-cut algorithm. In particular, in Section 3.6.2 we discuss the use of controlling parameters that we have implemented in our lazy constraint and user cut callback functions, in Section 3.6.3 we present a primal heuristic to provide feasible solutions to the branch-and-cut algorithm and in Section 3.6.4 we discuss the use of symmetry-breaking constraints.

3.6.1 The underlying formulation

The branch-and-cut algorithm is based on the MC formulation presented in Section 3.5.1. For clarity we rewrite it here:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1.1}$$

$$\text{subject to:} \sum_{j\in C} x_{dj} = 1 \qquad \forall d \in D \tag{1.2}$$

$$\sum_{j\in C} x_{jd} = 1 \qquad \forall d \in D \tag{1.3}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in C \tag{1.4}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in C \tag{1.5}$$

$$x(D \cup S', S) \geq 1 \qquad \forall S \subset C:\ 2 \leq |S| \leq |C| - |D| \tag{2.2}$$

$$x(S', d) + x(S', S) + x(d, S) \geq 1 \qquad \forall d \in D,\ \forall S \subset C \tag{2.13}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A. \tag{1.8}$$

More precisely, we give the (incomplete) formulation comprised of (1.1)–(1.5) and (1.8) to the CPLEX solver and then, whenever necessary, we add violated subtour elimination con-

straints (2.2) and/or violated 1-MCC inequalities (2.13) by using a lazy constraint callback function. To do so we use the separation algorithms 3.2 and 3.4, respectively, which are exact separation algorithms for integer points.

In the cutting plane phase of the branch-and-cut algorithm we also define a user cut callback function that searches for violated subtour elimination constraints (2.2) and/or violated 1-MCC inequalities (2.13) for fractional points. The separation algorithms we use are algorithms 3.1 and 3.2 for the subtour elimination constraints (2.2) and algorithms 3.3 and 3.4 for the 1-MCC inequalities (2.13).

Finally, we also search for violated valid inequalities for fractional points with the use of a user cut callback function, since the results analyzed in Section 3.5.2 showed that using valid inequalities helped decrease the optimal solution times. More precisely, we use as valid inequalities the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ by using the heuristic separation algorithm 3.6 every time a violated 1-MCC inequality (2.13) is found. In addition, we use the directed chain-barring constraints (2.17)–(2.18) as valid inequalities but we only separate them, by using the heuristic separation algorithm 3.9, at the root node of the branch-and-bound tree and only for symmetric cost instances.

3.6.2 Parameters for lazy constraint/user cut callback functions

As we explained at the end of Sections 3.2.2 and 3.2.3, a good implementation of a lazy constraint callback function or a user cut callback function is very important. Recall that the CPLEX solver has no control over the code in a callback function and, thus, the user must ensure that the code is efficient and produces the desired outcome. In this section we show some techniques that we used to guarantee the efficiency of our lazy constraint and user cut callback functions, namely regarding the order in which the separation algorithms are used and the control of the number of violated inequalities added before re-optimizing.

For controlling the separation process in our lazy constraint/user cut callback functions, we defined what we called a separation priority list. Essentially, the separation algorithms are put on a list in the order in which they should be used. With respect to the lazy constraints, the separation priority list is comprised of, in order, algorithm 3.2 for the subtour elimination constraints $x(D \cup S', S) \geq 1$ (2.2) and algorithm 3.4 for the 1-MCC inequalities $x(S', d)+x(S', S)+x(d, S) \geq 1$ (2.13). If the reversed order had been used no difference would have been observed, given that the lazy constraint callback functions are called infrequently.

Regarding user cuts, and for asymmetric instances, the separation priority list is comprised of, in order, algorithm 3.2 for the subtour elimination constraints (2.2), algorithm 3.4 for the 1-MCC inequalities (2.13), algorithm 3.1 for the subtour elimination constraints (2.2) and algorithm 3.3 for the 1-MCC inequalities (2.13). As for user cuts in symmetric instances, the separation priority list is comprised of, in order, algorithm 3.2 for the subtour elimination con-

straints (2.2), algorithm 3.4 for the 1-MCC inequalities (2.13), algorithm 3.9 for the directed chain-barring constraints (2.17)–(2.18), algorithm 3.1 for the subtour elimination constraints (2.2) and algorithm 3.3 for the 1-MCC inequalities (2.13). Note that we mentioned that we add the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ as valid inequalities in the branch-and-cut algorithm, however, the way in which we implemented algorithm 3.6 is such that it is incorporated into algorithms 3.3 and 3.4 and, therefore, the k-MCC inequalities (2.14) for $k \geq 2$ are separated alongside the 1-MCC inequalities (2.13). The order of the separation algorithms for user cut callback functions is of great importance since these functions are called very frequently during the branch-and-cut algorithm. The most important aspect that we need to ensure is that heuristic separation algorithms are called before exact separation algorithms. Afterwards, the choice of separating subtour elimination constraints before path elimination constraints was based on some initial computational testing which showed that, on average, this order performed slightly better.

We also introduce two control parameters. Firstly, we added a parameter to decide how many lazy constraints or user cuts should be added in total before re-optimizing. This is typical and straightforward and the purpose it to prevent the addition of too many cuts, which is known to hinder the overall process. We use the value of 1 for lazy constraints and the value of 20 for user cuts, that is, the lazy constraint separation process stops as soon as it finds one violated inequality, since that is all which is required to identify the integer point as unfeasible, and the user cut separation process stops whenever a combined total of 20 user cuts have been found, which seemed to be an effective value in our initial testing.

Secondly, we added another parameter for user cuts that decides whether or not the separation process should continue to the next algorithm in the separation priority list. The way in which this parameter works is the following. If in the current separation algorithm we add at least a given number of violated user cuts, then the separation process stops, that is, all algorithms which have a lower priority in the separation priority list are ignored. After some initial testing we decided to use the value 5, which is 25% of the maximum user cuts allowed of 20. The idea for this controlling parameter is that if a good percentage of user cuts are from a specific type (in this case, 25%), then it is preferable to re-optimize and restart the separation process so we can retry that specific separation algorithm sooner. This is also very helpful whenever we are using heuristic separation algorithms, which is the case, since if a good number of heuristic cuts are found then the exact separation algorithms with a lower priority are skipped.

3.6.3 A primal heuristic

During the branch-and-cut algorithm we regularly use a heuristic procedure to try to improve the current best known solution. This heuristic depends on whether the solution of the linear programming relaxation of the branch-and-bound node is fractional or integer, and it is imple-

mented in a heuristic callback function.

The heuristic procedure works as follows. Given a non-negative cost function c', we start by sorting the outgoing arcs from the depots in ascending order of cost and then choosing one arc for each depot, starting with the one with the lowest cost and without repeating client nodes. Then, by using a nearest neighbor criterion we insert the remaining client nodes in the best possible circuit until we form $|D|$ disjoint circuits, one for each depot. To further optimize these circuits we apply local search operators that first swap nodes inside the circuits, then move nodes from their current circuit to a different circuit and, finally, swap nodes inside the circuits again. If the heuristic is applied by using the original cost function c, the solutions obtained are usually too costly. However, we designed the heuristic with the purpose of using the information provided by the linear programming relaxation. More precisely, given fractional values x^*, we modify the cost of an arc (i, j) to $c_{ij} \times (1 - x^*_{ij})$. The reasoning for this modified cost function is that arcs for which the linear programming relaxation value is close to 1 will have a lower cost and hence have a higher probability of being chosen in the constructive part of the heuristic, that is, the feasible solution obtained before applying local search will be as close as possible to the current linear programming relaxation solution.

For nodes of the branch-and-bound tree in which an integer solution was found, the heuristic does not require the constructive part described above and only applies the local search operators to the solution found, namely it first swaps nodes inside the circuits, then moves nodes from their circuit to a different circuit and finally swaps nodes inside the circuits again.

The heuristic procedure is applied every time a new integer solution is found and every time it is called in the root node. For the remaining nodes in which the linear programming relaxation solution is fractional, the heuristic procedure is applied, for the first 250 branch-and-bound nodes, every 5 nodes and, after this, it is applied every 10 nodes.

3.6.4 Symmetry-breaking constraints for symmetric instances

If the cost function is a symmetric cost function, then a symmetry problem arises due to the use of a directed graph based formulation. In particular, any circuit with two or more clients and its reverse have the exact same cost and, therefore, represent equivalent solutions even if they are structurally different. This can pose a significant problem since any solution comprised of m circuits with two or more client nodes can be represented in 2^m different ways by combining the two possible orientations for the m circuits with two or more client nodes. This negatively influences the branch-and-cut algorithm since time may be wasted solving nodes which are equivalent.

In order to break these symmetries, observe that in a given circuit with two or more client nodes, one of the two client nodes adjacent to the depot has a smaller index than the other. Therefore, we can enforce that the first client node visited after the depot is given a lower index

than the client node visited just before the depot. For example, for the circuits $c_1 = (2, 5, 7, 4, 2)$ and $c_2 = (2, 4, 7, 5, 2)$, assuming that 2 is the depot node, the circuit c_1 would be considered unfeasible, leaving circuit c_2 to be one of the possible circuits to be part of a solution. More formally, we add the following constraints to the MC formulation for symmetric cost instances:

$$\sum_{j \in C,\, j \geq i} x_{jd} \geq x_{di} \qquad \forall d \in D,\ \forall i \in C. \tag{3.2}$$

Constraints (3.2) state that if a client $i \in C$ is visited immediately after a depot $d \in D$, then the client $j \in C$ visited just before d should be such that $j \geq i$. Notice that the case $j = i$ is required in order to avoid cutting-off solutions that are composed of two-node circuits.

We will not be providing any computational results to assess the benefit of using these constraints. Initial testing showed that the improvements are not significant, which may be attributed to the fact that most solutions have many circuits with only one depot and one client, however, and on average, the computational times are slightly reduced by the use of the symmetry-breaking constraints (3.2) and, thus, we will use these constraints in the branch-and-cut algorithm.

3.7 Computational experiment

In this section we present computational results to assess the performance of the branch-and-cut algorithm described in Section 3.6 in solving the test instances described in Section 3.4 to optimality. The instances in set T are too small and are not considered here. In addition, we also provide results regarding the linear programming relaxation value, namely we show the value of the linear programming relaxation of the underlying formulation by excluding all of CPLEX's general purpose cuts as well as preprocessing. The idea is to show how close the linear programming relaxation value is to the optimal value, however, note that this is not necessarily the same value as the one which is obtained in the root node of the branch-and-bound tree when solving the problem to optimality.

We divide this study in two parts, one concerning the asymmetric instances in Section 3.7.1, and the other the symmetric instances in Section 3.7.2. We also provide an additional set of computational results in Section 3.7.3 to compare the branch-and-cut algorithm to a different one in which we do not separate the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$.

3.7.1 Results for asymmetric instances

In this first part we present the results of applying the branch-and-cut algorithm to the asymmetric instances, that is, the instances in set C described in Section 3.4.

Table 3.3: Optimal solution results for asymmetric instances (1 of 2)

Name	OPT	t (s)	B&B	#SEC	#1MCC	#kMCC
bgs-100-5-1a	912	6	129	578	54	0
bgs-100-10-1a	921	2	15	122	38	2
bgs-100-20-1a	979	5	50	105	240	61
bgs-100-5-2a	980	4	90	480	53	2
bgs-100-10-2a	998	4	59	169	77	14
bgs-100-20-2a	1066	3	15	45	105	76
bgs-100-5-3a	928	3	49	278	20	0
bgs-100-10-3a	955	18	323	645	438	85
bgs-100-20-3a	994	7	50	176	265	188
bgs-200-10-1a	1325	14	30	333	66	1
bgs-200-20-1a	1361	159	432	738	887	94
bgs-200-40-1a	1504	250	415	196	1575	963
bgs-200-10-2a	1323	468	1523	1857	1090	202
bgs-200-20-2a	1355	99	150	329	858	177
bgs-200-40-2a	1475	2458	4599	513	4711	5405
bgs-200-10-3a	1395	70	258	824	164	3
bgs-200-20-3a	1422	751	1618	1503	3258	875
bgs-200-40-3a	1507	9607	10397	1110	14466	6793

Tables 3.3 and 3.4 show the results with respect to obtaining the optimal solution, with Table 3.3 focusing on the instances with up to 200 clients and up to 40 depots and Table 3.4 on the instances with 300 clients and up to 60 depots. Both tables have the following format. The first column indicates the name of the instance. The other six columns show, respectively, the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit was reached (OPT) and the corresponding time taken in seconds (t), the number of branch-and-bound nodes explored (B&B), the number of violated subtour elimination constraints $x(D \cup S', S) \geq 1$ (2.2) found (#SEC), the number of violated 1-MCC inequalities $x(S', d)+x(S', S)+x(d, S) \geq 1$ (2.13) found (#1MCC) and the number of violated k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ found (#kMCC). We imposed a time limit of 10800 seconds (three hours).

Tables 3.5 and 3.6 show the results with respect to obtaining the linear programming relaxation value, with Table 3.5 focusing on the instances with up to 200 clients and up to 40 depots and Table 3.6 on the instances with 300 clients and up to 60 depots. Both tables have the following format. The first column indicates the name of the instance and the second column indicates its optimal value (OPT) taken from either Table 3.3 or Table 3.4. The following six columns show, respectively, the linear programming relaxation value (LP), the percentage of

Table 3.4: Optimal solution results for asymmetric instances (2 of 2)

Name	OPT	t (s)	B&B	#SEC	#1MCC	#kMCC
bgs-300-10-1a	[1648, 1657]	10800*	7296	9388	1189	83
bgs-300-20-1a	[1668, 1698]	10800*	3331	7008	8979	965
bgs-300-30-1a	[1697, 1714]	10800*	2625	2708	11414	2537
bgs-300-40-1a	[1724, 1736]	10800*	3377	1492	9913	2999
bgs-300-60-1a	1820	8760	3819	721	5103	3939
bgs-300-10-2a	1612	492	334	2435	45	0
bgs-300-20-2a	1633	5617	4148	2900	2610	286
bgs-300-30-2a	1665	1626	1027	1504	1082	219
bgs-300-40-2a	1696	3955	861	1441	4812	1612
bgs-300-60-2a	1789	7146	1724	1149	5995	4521
bgs-300-10-3a	[1610, 1617]	10800*	6959	6055	1560	58
bgs-300-20-3a	[1623, 1645]	10800*	2100	2672	10824	948
bgs-300-30-3a	[1648, 1672]	10800*	2499	2143	11465	2398
bgs-300-40-3a	[1698, 1714]	10800*	2204	2184	11618	4138
bgs-300-60-3a	[1777, 1783]	10800*	3269	764	5776	4339

*Not solved to optimality within the limit of three hours.

gap between the linear programming relaxation value and the best known upper bound (gap), the time taken to obtain the linear programming relaxation value (t_L) in seconds, the number of violated subtour elimination constraints (2.2) found (#SEC), the number of violated 1-MCC inequalities (2.13) found (#1MCC) and the number of violated k-MCC inequalities (2.14) for $k \geq 2$ found (#kMCC).

First we analyze the results in Table 3.3. Regarding the asymmetric instances with 100 clients and up to 20 depots, the branch-and-cut algorithm was able to solve all nine instances within at most 18 seconds. Since the computational times are negligible, it is not clear how the number of depots influences the solution times. We can also see that the number of violated multi-cut constraints found generally increases with the number of depots. In particular, the number of violated k-MCC inequalities (2.14) for $k \geq 2$ found corresponds to a relatively high percentage of the total number of violated inequalities found for instances with 20 depots. For example, in instances bgs-100-20-2a and bgs-100-20-3a they are roughly one third of the total number of violated inequalities found. This is also indicative that the heuristic separation algorithm 3.6 that we proposed is effective at finding violated k-MCC inequalities (2.14) for $k \geq 2$.

Still in Table 3.3, the results show that the instances with 200 clients and up to 40 depots are harder to solve than the previous ones, however, the branch-and-cut algorithm was still able to solve all nine instances to optimality within the time limit of three hours. For this set, the results show that instances with more depots are, in general, harder to solve. For example, instance bgs-

Table 3.5: Linear programming relaxation results for asymmetric instances (1 of 2)

Name	OPT	LP	gap (%)	t_L (s)	#SEC	#1MCC	#kMCC
bgs-100-5-1a	912	898.349	1.50	0	92	2	0
bgs-100-10-1a	921	911.38	1.04	0	46	14	3
bgs-100-20-1a	979	969.67	0.95	0	38	65	19
bgs-100-5-2a	980	967.948	1.23	0	120	3	0
bgs-100-10-2a	998	988.778	0.92	0	101	11	0
bgs-100-20-2a	1066	1055.8	0.96	0	43	43	35
bgs-100-5-3a	928	917.25	1.16	0	123	0	0
bgs-100-10-3a	955	939.458	1.63	0	96	27	19
bgs-100-20-3a	994	983.065	1.10	1	96	96	95
bgs-200-10-1a	1325	1319.7	0.40	3	202	28	0
bgs-200-20-1a	1361	1349	0.88	4	127	60	14
bgs-200-40-1a	1504	1492.61	0.76	9	77	164	122
bgs-200-10-2a	1323	1308.76	1.08	3	193	23	0
bgs-200-20-2a	1355	1344.02	0.81	3	184	48	9
bgs-200-40-2a	1475	1452.67	1.51	24	73	218	237
bgs-200-10-3a	1395	1380.69	1.03	1	136	2	0
bgs-200-20-3a	1422	1407.13	1.05	2	86	29	9
bgs-200-40-3a	1507	1489.21	1.18	19	102	169	285

200-20-3a was solved in 751 seconds whereas for instance bgs-200-40-3a the solution time was of 9607 seconds. In addition, we can see that the number of violated multi-cut constraints found can increase significantly with the number of depots. An interesting case is that of instance bgs-200-40-2a where the number of violated k-MCC inequalities (2.14) for $k \geq 2$ found was about 50% of the total number of violated inequalities found.

As for the results in Table 3.4, regarding the instances with 300 clients and up to 60 depots, the branch-and-cut algorithm was only able to solve 6 out of 15 instances. These results are a clear evidence of the limit of capability of the branch-and-cut algorithm in order to solve asymmetric instances. Interestingly, in the instance set 300-xx-1, the branch-and-cut algorithm was able to solve the instance with 60 depots but not the other ones and, in the instance set 300-xx-3, no instance was solved, however, the instance with 60 depots ended, after three hours, with a smaller interval of lower and upper bounds when compared to the instances with between 20 and 40 depots. These results allow us to conclude that the limitation of the branch-and-cut algorithm is more likely due to the ratio between the number of clients and depots rather than the total number of clients and depots individually.

Focusing now on the results regarding the linear programming relaxation values of Tables 3.5 and 3.6, we first observe that, for unsolved instances, the value reported on the gap column

Table 3.6: Linear programming relaxation results for asymmetric instances (2 of 2)

Name	OPT	LP	gap (%)	t_L (s)	#SEC	#1MCC	#kMCC
bgs-300-10-1a	[1648, 1657]	1633.52	1.42	13	290	16	1
bgs-300-20-1a	[1668, 1698]	1656.83	2.42	25	249	88	2
bgs-300-30-1a	[1697, 1714]	1681.68	1.89	50	193	273	54
bgs-300-40-1a	[1724, 1736]	1710.1	1.49	68	147	321	97
bgs-300-60-1a	1820	1803.32	0.92	296	126	704	656
bgs-300-10-2a	1612	1599.05	0.80	4	212	1	0
bgs-300-20-2a	1633	1622.34	0.65	10	248	18	0
bgs-300-30-2a	1665	1655.84	0.55	15	191	30	7
bgs-300-40-2a	1696	1688.95	0.42	33	143	135	73
bgs-300-60-2a	1789	1777.79	0.63	243	140	672	352
bgs-300-10-3a	[1610, 1617]	1595.79	1.31	11	261	2	0
bgs-300-20-3a	[1623, 1645]	1609.93	2.13	15	225	44	8
bgs-300-30-3a	[1648, 1672]	1634.4	2.25	30	179	102	42
bgs-300-40-3a	[1698, 1714]	1687.62	1.54	73	239	238	129
bgs-300-60-3a	[1777, 1783]	1764.94	1.01	151	126	407	430

is an upper bound on the gap value if it were calculated by using the actual optimal value. In addition, as we mentioned before, the lower bound determined on the root node when solving the problem to optimality is not necessarily the same as the linear programming relaxation value that we report in these tables, since CPLEX's general purpose cuts and preprocessing are not deactivated for the former. As for the results, and considering first only solved instances, we can see that there was a maximum gap of 1.63% for the instances with 100 clients and up to 20 depots, 1.51% for the instances with 200 clients and up to 40 depots, and 0.92% for the instances with 300 clients and up to 60 depots. These values are very satisfactory and they help explain why the particular instances with 300 clients and up to 60 depots were solved while the ones with less depots and the same 300 clients were not. As for the unsolved instances in Table 3.6, the maximum reported gap was of 2.42% which, considering that it is an upper bound on the gap calculated with the unknown optimal solution, it is still a very satisfactory value. Finally, we observe that the number of violated inequalities of each type follows the same pattern as in the case of Tables 3.3 and 3.4, which were with respect to obtaining the optimal solution.

3.7.2 Results for symmetric instances

In this second part we present the results of applying the branch-and-cut algorithm for the symmetric instances, which are the instances in sets A and B described in Section 3.4.

Tables 3.7, 3.8 and 3.9 show the results with respect to obtaining the optimal solution, with

Table 3.7: Optimal solution results for symmetric instances (1 of 3)

Name	OPT	t (s)	B&B	#SEC	#1MCC	#kMCC	#CB
100-5-1	38116	45	1480	1863	11	0	0
100-10-1	41991	6	76	253	56	8	9
100-5-2	34018	13	367	1187	29	0	0
100-10-2	39126	2	18	230	7	1	5
100-5-3	33024	9	192	606	7	0	1
100-10-3	33719	13	387	664	89	1	2
200-10-1	53739	1732	10062	2586	616	8	4
200-10-2	47441	3764	23926	3051	155	2	2
200-10-3	47828	433	2033	2056	100	8	3

Table 3.7 focusing on the symmetric benchmark location-routing problem instances of set A, Table 3.8 on the randomly generated instances of set B with up to 200 clients and up to 40 depots and Table 3.9 on the randomly generated instances of set B with 300 clients and up to 60 depots. All three tables have the following format. The first column indicates the name of the instance. The other seven columns show, respectively, the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit was reached (OPT) and the corresponding time taken in seconds (t), the number of branch-and-bound nodes explored (B&B), the number of violated subtour elimination constraints $x(D \cup S', S) \geq 1$ (2.2) found (#SEC), the number of violated 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) found (#1MCC), the number of violated k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ found (#kMCC) and the number of violated directed chain-barring constraints (2.17)–(2.18) found (#CB). We again imposed a time limit of 10800 seconds (three hours).

Tables 3.10, 3.11 and 3.12 show the results with respect to obtaining the linear programming relaxation value, with Table 3.10 focusing on the symmetric benchmark location-routing problem instances of set A, Table 3.11 on the randomly generated instances of set B with up to 200 clients and up to 40 depots and Table 3.12 on the randomly generated instances of set B with 300 clients and up to 60 depots. All three tables have the following format. The first column indicates the name of the instance and the second column indicates its optimal value (OPT) taken from either Table 3.7, Table 3.8 or Table 3.9. The following seven columns show, respectively, the linear programming relaxation value (LP), the percentage of gap between the linear programming relaxation value and the best known upper bound (gap), the time taken to obtain the linear programming relaxation value (t_L) in seconds, the number of violated subtour elimination constraints (2.2) found (#SEC), the number of violated 1-MCC inequalities (2.13) found (#1MCC), the number of violated k-MCC inequalities (2.14) for $k \geq 2$ found (#kMCC) and the number of violated directed chain-barring constraints (2.17)–(2.18) found (#CB).

Table 3.8: Optimal solution results for symmetric instances (2 of 3)

Name	OPT	t (s)	B&B	#SEC	#1MCC	#kMCC	#CB
bgs-100-5-1	1587	1	0	179	3	0	2
bgs-100-10-1	1597	4	28	324	38	10	3
bgs-100-20-1	1687	7	74	301	84	17	13
bgs-100-5-2	1744	1	0	208	7	0	4
bgs-100-10-2	1777	2	5	208	21	3	5
bgs-100-20-2	1835	11	120	217	211	39	12
bgs-100-5-3	1553	28	765	1127	3	0	1
bgs-100-10-3	1558	13	302	439	47	5	1
bgs-100-20-3	1604	1	0	100	7	0	6
bgs-200-10-1	2301	121	395	1011	143	5	0
bgs-200-20-1	2332	536	1891	1301	651	29	7
bgs-200-40-1	2417	376	711	1014	1017	115	13
bgs-200-10-2	2236	700	3097	2433	364	13	5
bgs-200-20-2	2291	896	3655	1390	1134	304	5
bgs-200-40-2	2433	6783	18536	1074	6107	2388	26
bgs-200-10-3	2316	3476	15840	4493	802	60	3
bgs-200-20-3	[2345, 2346]	10800*	36617	2424	4553	760	10
bgs-200-40-3	2417	1934	5645	775	2630	981	13

*Not solved to optimality within the limit of three hours.

We start by analyzing the results in Table 3.7 regarding the benchmark location-routing problem instances. The results show that the branch-and-cut algorithm was able to solve the instances with 100 clients in at most 45 seconds and the instances with 200 clients in at most around one hour. The number of depots in these instances is fairly low and without much variation and, thus, these results do not allow us to perform a meaningful analysis concerning this particular aspect. Observe that the number of violated multi-cut constraints found in these instances is also low, which we believe is related to the small number of depots. In addition, notice that the number of violated directed chain-barring constraints (2.17)–(2.18) found is significant in most instances considering that they are only separated in the root node.

Concerning the results of Table 3.8, which are with respect to the randomly generated instances with up to 200 clients and up to 40 depots, we see that the branch-and-cut algorithm was able to solve all instances with 100 clients in at most 28 seconds and all instances with 200 clients in at most around two hours, except instance bgs-200-20-3 which ended with a lower bound of 2345 and an upper bound of 2346. In addition, these instances seem to be of the same type of difficulty when compared to the benchmark location-routing problem instances of Table 3.7, however, when compared to equal-sized asymmetric instances, in general they take slightly

Table 3.9: Optimal solution results for symmetric instances (3 of 3)

Name	OPT	t (s)	B&B	#SEC	#1MCC	#kMCC	#CB
bgs-300-10-1	[2723, 2737]	10800*	7171	8070	556	27	1
bgs-300-20-1	[2738, 2779]	10800*	6182	5994	1922	100	4
bgs-300-30-1	[2777, 2802]	10800*	5150	3070	4424	319	3
bgs-300-40-1	[2826, 2847]	10800*	3551	1619	7424	831	7
bgs-300-60-1	[2920, 2977]	10800*	1692	1785	8176	1518	17
bgs-300-10-2	[2755, 2779]	10800*	7545	6175	543	8	0
bgs-300-20-2	[2762, 2779]	10800*	5581	3417	3230	136	3
bgs-300-30-2	[2815, 2820]	10800*	4088	2500	4862	944	6
bgs-300-40-2	[2865, 2892]	10800*	2379	1852	7141	2318	12
bgs-300-60-2	[2965, 2976]	10800*	2939	869	7248	2380	30
bgs-300-10-3	[2727, 2745]	10800*	7483	6667	323	3	0
bgs-300-20-3	[2757, 2763]	10800*	8375	2060	1172	80	4
bgs-300-30-3	2781	9017	7874	1646	1885	82	10
bgs-300-40-3	[2817, 2839]	10800*	4351	1897	4456	775	11
bgs-300-60-3	[2870, 2876]	10800*	4064	1229	3452	567	22

*Not solved to optimality within the limit of three hours.

longer to solve and the number of violated multi-cut constraints found is lower. We believe that there is an explanation for this situation and we will discuss it after analyzing the next table of results. Finally, we can again see that the number of violated directed chain-barring constraints (2.17)–(2.18) found is significant in most instances.

We now analyze the results regarding the randomly generated instances with 300 clients and up to 60 depots reported in Table 3.9. These results show that only one instance out of 15 was solved within the time limit, which was instance bgs-300-30-3 with a solution time of 9017 seconds. When compared to the results for asymmetric instances, and also taking into the account the analysis of the results of Table 3.8, it becomes clear that the branch-and-cut algorithm is not as effective for symmetric instances as for asymmetric instances. The explanation for this situation is based on the result of Proposition 3 which also applies to the multi-cut constraints and which essentially states that the linear programming relaxation values for symmetric instances when using the multi-cut constraints are expected to be worse overall when compared to asymmetric instances. More precisely, the cutting plane phase of the branch-and-cut algorithm is able to provide higher linear programming relaxation values for asymmetric instances than for symmetric instances and, thus, the overall branch-and-cut algorithm for asymmetric instances is more effective. This is also evident from the number of violated k-MCC inequalities (2.14) found, including for $k = 1$, which is lower, in general, for symmetric instances.

Focusing now on the results regarding the linear programming relaxation values of Tables

Table 3.10: Linear programming relaxation results for symmetric instances (1 of 3)

Name	OPT	LP	gap (%)	t_L (s)	#SEC	#1MCC	#kMCC	#CB
100-5-1	38116	37543.7	1.50	0	223	0	0	2
100-10-1	41991	41178.8	1.93	0	140	6	0	5
100-5-2	34018	33681.1	0.99	0	165	0	0	5
100-10-2	39126	38603.5	1.34	0	183	0	0	6
100-5-3	33024	32622.5	1.22	0	309	0	0	2
100-10-3	33719	33024	2.06	0	227	0	0	2
200-10-1	53739	52847.8	1.66	3	309	0	0	7
200-10-2	47441	46758.5	1.44	3	332	0	0	0
200-10-3	47828	47425.3	0.84	2	325	0	2	2

3.10, 3.11 and 3.12, we can see that the linear programming relaxation values are, on average, worse than for asymmetric instances, as expected due to the result of Proposition 3. Nevertheless, the maximum gap for the benchmark location-routing problem instances was 2.06%, for the randomly generated instances with 100 clients and up to 20 depots it was 2.89%, for the randomly generated instances with 200 clients and up to 40 depots the maximum gap it was 2.37%, and concerning the randomly generated instances with 300 clients and up to 60 it was 2.98%, which are still satisfactory values. Observe also that the violated constraints found when obtaining the linear programming relaxation values consist mostly of subtour elimination constraints (2.2) and a few directed chain-barring constraints (2.17)–(2.18).

3.7.3 Evaluating the effectiveness of the generalized multi-cut constraints

In Section 3.5.2 we presented a preliminary set of computational results that showed that the separation of the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ by using the heuristic separation algorithm 3.6 produced slight decreases in the overall time taken to obtain the optimal solution. That comparison, however, was based on a default branch-and-cut algorithm. In this section we present a similar comparison by using the branch-and-cut algorithm described in Section 3.6. More precisely, we compare two branch-and-cut algorithms. The first one, which we will denote by B&C$_1$ in the remainder of this section, is the branch-and-cut algorithm described in Section 3.6 used to obtain the results in the two previous sections. The second one, denoted by B&C$_2$, is similar to the branch-and-cut algorithm B&C$_1$ but in which we do not separate the k-MCC inequalities (2.14) for $k \geq 2$.

Table 3.13 presents the comparison results between the two branch-and-cut algorithms for the asymmetric and the symmetric randomly generated bgs instances with 200 clients and up to 40 depots with the following format. The first column indicates the name of the instance. The remaining six columns are divided into two parts, each with three columns corresponding

Table 3.11: Linear programming relaxation results for symmetric instances (2 of 3)

Name	OPT	LP	gap (%)	t_L (s)	#SEC	#1MCC	#kMCC	#CB
bgs-100-5-1	1587	1578.5	0.54	0	181	0	0	1
bgs-100-10-1	1597	1580	1.06	0	217	4	0	6
bgs-100-20-1	1687	1660.5	1.57	0	154	2	0	15
bgs-100-5-2	1744	1728.5	0.89	0	226	0	0	1
bgs-100-10-2	1777	1755.19	1.23	0	186	1	0	12
bgs-100-20-2	1835	1792	2.34	0	92	1	0	14
bgs-100-5-3	1553	1517	2.32	0	107	0	0	2
bgs-100-10-3	1558	1513	2.89	0	99	1	0	1
bgs-100-20-3	1604	1568	2.24	0	116	0	0	15
bgs-200-10-1	2301	2290.85	0.44	3	359	0	0	3
bgs-200-20-1	2332	2306.22	1.11	4	264	2	0	19
bgs-200-40-1	2417	2381.6	1.46	4	180	5	1	17
bgs-200-10-2	2236	2206.2	1.33	2	236	0	0	4
bgs-200-20-2	2291	2250.67	1.76	3	237	1	0	13
bgs-200-40-2	2433	2384	2.01	4	166	9	0	28
bgs-200-10-3	2316	2280.25	1.54	3	209	5	1	3
bgs-200-20-3	[2345, 2346]	2299.75	1.97	5	285	0	0	6
bgs-200-40-3	2417	2359.77	2.37	3	144	3	0	17

to, respectively, the linear programming relaxation value (LP), the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit of 10800 seconds was reached (OPT) and the respective time taken (t) in seconds. The first part corresponds to the branch-and-cut algorithm described in Section 3.6, that is, the branch-and-cut algorithm B&C$_1$, and the second part corresponds to the branch-and-cut algorithm in which the k-MCC inequalities (2.14) for $k \geq 2$ are not separated, that is, the branch-and-cut algorithm B&C$_2$. The results for the branch-and-cut algorithm B&C$_1$ were taken from previous tables, namely Tables 3.3 and 3.5 for the asymmetric instances and Tables 3.8 and 3.11 for the symmetric instances.

We will divide our analysis between the asymmetric and the symmetric instances, which correspond to, respectively, the first nine instances and the last nine instances. Regarding the former, observe that the branch-and-cut algorithm B&C$_1$ was able to solve all nine instances in an average of 1542 seconds, whereas the branch-and-cut algorithm B&C$_2$ could only solve seven out of nine instances and with an average running time of 2636 seconds. However, if we consider only the instances with 10 or 20 depots, then the branch-and-cut algorithm B&C$_1$ solves the six instances in an average of 260 seconds, while the branch-and-cut algorithm B&C$_2$ in an average of 230 seconds. This shows that the benefit of the use of the k-MCC inequalities (2.14) for $k \geq 2$ is only clearly evident in instances with more depots. Note also that for the

Table 3.12: Linear programming relaxation results for symmetric instances (3 of 3)

Name	OPT	LP	gap (%)	t_L (s)	#SEC	#1MCC	#kMCC	#CB
bgs-300-10-1	[2723, 2737]	2700.5	1.33	14	370	0	0	4
bgs-300-20-1	[2738, 2779]	2718.56	2.17	12	463	0	0	12
bgs-300-30-1	[2777, 2802]	2755.1	1.67	14	399	0	0	8
bgs-300-40-1	[2826, 2847]	2798.55	1.70	16	277	3	1	21
bgs-300-60-1	[2920, 2977]	2888.28	2.98	26	319	10	1	32
bgs-300-10-2	[2755, 2779]	2724.63	1.96	14	412	0	0	5
bgs-300-20-2	[2762, 2779]	2738.22	1.47	13	358	0	0	4
bgs-300-30-2	[2815, 2820]	2781.28	1.37	29	399	7	0	16
bgs-300-40-2	[2865, 2892]	2828.03	2.21	26	350	8	0	33
bgs-300-60-2	[2965, 2976]	2918.46	1.93	22	248	9	0	33
bgs-300-10-3	[2727, 2745]	2707.5	1.37	18	626	0	0	1
bgs-300-20-3	[2757, 2763]	2727.25	1.29	20	517	0	1	7
bgs-300-30-3	2781	2743.33	1.35	27	417	4	0	10
bgs-300-40-3	[2817, 2839]	2776.69	2.19	17	296	4	0	16
bgs-300-60-3	[2870, 2876]	2830.42	1.58	19	268	11	1	19

instances with 10 or 20 depots, the difference in the linear programming relaxation value is negligible, whereas for the instances with 40 depots the difference is more noticeable. This also helps to explain why the use of the k-MCC inequalities (2.14) for $k \geq 2$ is more beneficial for the instances with more depots.

As for the symmetric instances, both branch-and-cut algorithms were able to solve eight out of nine instances within the time limit, with an average running time of 2847 seconds and 2881 seconds for the branch-and-cut algorithm B&C$_1$ and the branch-and-cut algorithm B&C$_2$, respectively. In other words, the branch-and-cut algorithm B&C$_1$ was only slightly better than the branch-and-cut algorithm B&C$_2$ for symmetric instances. Observe that the linear programming relaxation values are the same for both cases due to the result of Proposition 3, which is one explanation for the similarity of the results obtained with both branch-and-cut algorithms.

The results show that the use of the k-MCC inequalities (2.14) for $k \geq 2$ is more beneficial for asymmetric instances with a considerable number of depots. For the remaining cases they provide only slightly lower average computational times. Thus, if one is able to find a straightforward adaptation of the k-MCC inequalities (2.14) for $k \geq 2$ in other problems and, more importantly, if one is able to find an effective heuristic separation algorithm for the more general multi-cut constraints, then it may be worth investigating their use in a branch-and-cut algorithm. However, if one is unable to do so, the expected performance of a branch-and-cut algorithm without the k-MCC inequalities (2.14) for $k \geq 2$ should not be substantially different when compared to a branch-and-cut algorithm with these inequalities.

Table 3.13: Evaluating the effectiveness of the generalized multi-cut constraints

	B&C$_1$			B&C$_2$		
Name	LP	OPT	t (s)	LP	OPT	t (s)
bgs-200-10-1a	1319.7	1325	14	1319.7	1325	14
bgs-200-20-1a	1349	1361	159	1348.97	1361	300
bgs-200-40-1a	1492.61	1504	250	1492.26	1504	745
bgs-200-10-2a	1308.76	1323	468	1308.76	1323	215
bgs-200-20-2a	1344.02	1355	99	1343.82	1355	105
bgs-200-40-2a	1452.67	1475	2458	1449.66	[1467, 1486]	10800*
bgs-200-10-3a	1380.69	1395	70	1380.69	1395	89
bgs-200-20-3a	1407.13	1422	751	1407.02	1422	659
bgs-200-40-3a	1489.21	1507	9607	1486.38	[1499, 1513]	10800*
bgs-200-10-1	2290.85	2301	121	2290.85	2301	155
bgs-200-20-1	2306.22	2332	536	2306.22	2332	541
bgs-200-40-1	2381.6	2417	376	2381.6	2417	236
bgs-200-10-2	2206.2	2236	700	2206.2	2236	1065
bgs-200-20-2	2250.67	2291	896	2250.67	2291	1284
bgs-200-40-2	2384	2433	6783	2384	[2427, 2436]	10800*
bgs-200-10-3	2280.25	2316	3476	2280.25	2316	3003
bgs-200-20-3	2299.75	[2345, 2346]	10800*	2299.75	2346	5941
bgs-200-40-3	2359.77	2417	1934	2359.77	2417	2911

*Not solved to optimality within the time limit of three hours

We believe that the adaptation of the k-MCC inequalities (2.14) for $k \geq 2$ to other problems may not be straightforward, which is the reason for establishing this comparison in this section. In fact, we will show in Chapter 6, when we discuss the Hamiltonian p-median problem in the second part of this dissertation, that, for that case, the adaptation of the k-MCC inequalities (2.14) for $k \geq 2$ is not clear.

3.8 Concluding remarks

In this chapter we proposed a branch-and-cut algorithm based on a formulation in the space of the arc variables x which uses the new multi-cut path elimination constraints, namely the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) presented in Chapter 2. Earlier theoretical investigations indicated that the new 1-MCC inequalities (2.13) could potentially be useful in practice for one important reason. As we saw in Chapter 2, the new constraints could be seen as equivalent in terms of linear programming relaxation to a compact system of inequalities that models path elimination constraints by using arc-depot assignment variables. This relationship

is based on the max-flow/min-cut theorem and, thus, we were able to devise a very efficient separation algorithm for the 1-MCC inequalities (2.13) which only requires solving as many max-flow problems as the number of depots. One drawback of these inequalities, however, is related to the fact that, if we consider the linear programming relaxation value given by a formulation with just the depot degree constraints (1.2)–(1.3), then by adding the 1-MCC inequalities (2.13) the linear programming relaxation value is unchanged if costs are symmetric, as a result of Proposition 3.

We also conducted some preliminary computational tests. In particular, we were interested in studying how the new 1-MCC inequalities (2.13) compared, in a branch-and-cut algorithm, to a previously know set of path elimination constraints, namely the so-called chain-barring constraints from which we derived the directed chain-barring constraints (2.17)–(2.18). These tests indicated that, for asymmetric instances, the 1-MCC inequalities (2.13) provide higher linear programming relaxation values, whereas, for symmetric instances, the directed chain-barring constraints (2.17)–(2.18) provide higher linear programming relaxation values. Nevertheless, we still believe that the 1-MCC inequalities (2.13) are overall the best option for symmetric instances. The reasoning, discussed in more detail in Section 3.5.1, is essentially that the directed chain-barring constraints (2.17)–(2.18) do not have a simultaneously exact (or near-exact) and efficient separation algorithm and, thus, we are required to resort to heuristic separation algorithms, has has already been observed in earlier studies (see Benavent & Martínez-Sykora 2013). Thus, by using the 1-MCC inequalities (2.13) as base path elimination constraints and the directed chain-barring constraints (2.17)–(2.18) as valid inequalities separated in a heuristic way, we believe that we are able to obtain better performances in a branch-and-cut algorithm.

In these preliminary computational tests we were also interested in studying the impact of using the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ along with the simpler 1-MCC inequalities (2.13). More precisely, we tested whether or not separating the k-MCC inequalities (2.14) for $k \geq 2$ by using the heuristic separation algorithm 3.6 alongside the 1-MCC inequalities (2.13) would lead to improvements over separating only the latter. These tests, and later ones conducted within the final setting of the branch-and-cut algorithm, showed that, on average, we obtained lower optimal solution times by separating the k-MCC inequalities (2.14) for $k \geq 2$, however, this reduction was barely noticeable for symmetric instances and not significant for asymmetric instances in the sense that the solution times were not reduced by an order of magnitude. Nevertheless, if one is able to find an efficient and effective algorithm for these constraints, such as algorithm 3.6, which is computationally fast and easy to implement, then it may be worth investigating their use in a branch-and-cut algorithm.

The performance of the branch-and-cut algorithm was overall satisfactory. For instances with 100 clients and up to 20 depots, the branch-and-cut algorithm reached the optimal solution in less than one minute. The optimal solution times were below the limit of three hours for

instances with 200 clients and up to 40 depots, both symmetric and asymmetric, which are instances of a considerable size, except in one instance. These times are the largest reported ones but in many of these instances the solution times were below or around 15 minutes, with a few between one hour and two hours. The bottleneck of the branch-and-cut algorithm was in the instances with 300 clients and up to 60 depots, where the algorithm could not solve the majority of the asymmetric instances and only solved one of the symmetric instances, however, an interesting conclusion is that the limit is not necessarily tied to the number of clients or the number of depots individually but rather to a ratio between the two numbers, since, in fact, instances with 300 clients and 60 depots proved to be more easily handled by the branch-and-cut algorithm than some instances with the same 300 clients but fewer depots.

We also conclude from the final results that on average, for equal-sized asymmetric and symmetric instances, the branch-and-cut algorithm usually solves the asymmetric instances faster. This is mainly due to the fact that, as we mentioned before, the multi-cut constraints on which the branch-and-cut algorithm heavily relies have the property described in Proposition 3. This difference between solving asymmetric and symmetric instances was expected even before any theoretical considerations. It is known from the literature regarding general routing problems that models based on undirected graphs are more suitable to solve symmetric cost instances, however, those models cannot be used for asymmetric cost instances whereas models based on directed graphs, such as the underlying formulation of our branch-and-cut algorithm, can be used for both cost structures. Nevertheless, the differences are not substantial and, in fact, the branch-and-cut algorithm is still effective for symmetric instances. We also observed that the linear programming relaxation values are always such that the maximum gap is slightly lower than 3% for the solved symmetric instances and lower than 2% for the solved asymmetric instances, which also explains why the algorithm performs slightly better for asymmetric instances. Regarding the violated valid inequalities found, we saw that the branch-and-cut algorithm identifies many more multi-cut constraints for asymmetric instances than for symmetric instances, which is also consistent with the result of Proposition 3.

While we are overall satisfied with the results, we believe that the branch-and-cut algorithm can be improved. From a theoretical point of view, the branch-and-cut algorithm could be improved by looking into using other valid inequalities to further enhance the cutting plane phase. Concerning more practical aspects, the performance of the branch-and-cut algorithm could be improved, for example, by finding other more effective heuristic separation algorithms for the inequalities in use, since max-flow/min-cut computations, despite being polynomial in time, are still slow and, thus, it is important to reduce them as much as possible.

Our study of the multi-depot routing problem continues in the following chapter by looking into additional theoretical aspects. More precisely, we will present different sets of path elimination constraints, which we will be comparing to the ones presented throughout Chapter 2. One

of those new sets will allow us to derive a new formulation for the multi-depot routing problem which provides linear programming relaxation values close to the optimal solution. Even though we will not be further improving the branch-and-cut algorithm presented in this chapter, the theoretical discussion of the following chapter is an important starting point for deriving additional constraints in the space of the x variables.

Chapter 4

Formulations using depot assignment variables

Contents

4.1 Introduction

In this chapter we study additional formulations for the multi-depot routing problem. In particular, we present several systems of path elimination constraints and both a theoretical and practical comparison of such systems, including a comparison to the path elimination constraints presented in Chapter 2, namely the multi-cut constraints and the compact systems of inequalities based on the arc-depot assignment variables z in addition to the adaptation of the so-called chain-barring constraints. We also study the set of constraints of the multi-depot routing problem in a broader sense, that is, we no longer focus solely on path elimination constraints but instead present a formulation which we will show implies most of the constraints presented in previous chapters and in the first part of this chapter. This formulation is based on depot assignment variables, both client-depot assignment variables and the arc-depot assignment variables z, and it models subtour elimination constraints and path elimination constraints simultaneously. We will show that this formulation provides linear programming relaxation values close to the optimal value with some computational experiments. Since this chapter is extensive and contains a lot of information, we will start by providing a summary of each section.

We start in Section 4.2 by presenting path elimination constraints based on variables that indicate which clients are assigned to which depots and which we call client-depot assignment variables. More precisely, we consider binary variables $v_d^i = 1$ if client $i \in C$ is assigned to or is in the circuit of depot $d \in D$, and $v_d^i = 0$ otherwise. These variables can also be viewed as precedence variables such as the ones used in the context of the precedence constrained (asymmetric) traveling salesman problem (see, e.g., Balas et al. 1995, Gouveia & Pires 1999, 2001, Gouveia & Pesneau 2006, Gouveia et al. 2018). This relationship stems from the fact that the client-depot assignment variables can be interpreted as precedence variables that indicate a precedence relationship between the depots and the clients. The additional information

provided by the precedence variables led to intuitive and useful constraints for the precedence constrained traveling salesman problem and, in fact, as we shall show, the client-depot assignment variables are also useful for modeling path elimination constraints in multi-depot routing problems.

We will start by considering a straightforward compact system of inequalities that can be used to model path elimination constraints. Then we show that this system of inequalities can be strengthened by using a similar approach to what has been done in the precedence constrained traveling salesman problem, namely we first strengthen it in a way that the system of inequalities we obtain remains a compact system of inequalities, and then we generalize this new system of inequalities to obtain a non-compact system of inequalities. Afterwards, we present enhancements that are directly motivated by the multi-depot routing problem, in particular by the fact that multiple depots exist, and we show that the original system of inequalities can be generalized in a way which is different from the generalization derived previously, with the resulting system of inequalities being an exponentially-sized set of constraints. Additionally, we show that the two different ways of generalizing the original system of inequalities can be combined. Then, we use the definition of the client-depot assignment variables v and of the arc-depot assignment variables z, originally presented in Section 2.3, to observe that we can establish a relationship between the two sets of variables and, from there, derive new systems of inequalities. Firstly, we use the relationship between the v and the z variables to show that the 3I^{++} system, which was presented in Section 2.3.3 and is based on the arc-depot assignment variables, implies two new sets of exponentially-many path elimination constraints. Secondly, we show how to generalize one of these new sets of path elimination constraints, again based on the relationship between the v and the z variables. Finally, we combine constraints presented in this section to derive new constraints in the space of the x variables.

In Section 4.3, we adapt the double multi-commodity network flow systems originally proposed by Wong (1980) for the traveling salesman problem. These double network flow systems use two sets of binary flow variables, namely binary variables $f_{pq}^{di} = 1$ if flow is sent from $d \in D$ to $i \in C$ via arc $(p, q) \in \left(A^C \setminus \{(i, j) : j \in C\}\right) \cup A_O^d$, and $f_{pq}^{di} = 0$ otherwise, and binary variables $g_{pq}^{di} = 1$ if flow is sent from $i \in C$ to $d \in D$ via arc $(p, q) \in \left(A^C \setminus \{(j, i) : j \in C\}\right) \cup A_I^d$, and $g_{pq}^{di} = 0$ otherwise, and the client-depot assignment v variables as auxiliary variables. We show that we obtain a valid set of path elimination constraints for the multi-depot routing problem by adding constraints that link these double network flow systems with the x variables, however, we also show that we can obtain an stronger system of inequalities if we combine the double network flow systems with the systems of inequalities based on the arc-depot assignment variables, by using stronger linking constraints between the f and g and the z variables. Additionally, we observe that this new system of inequalities models both subtour elimination constraints and path elimination constraints and, in fact, the formulation resulting from using

this system of inequalities is the formulation with the highest linear programming relaxation value that we will propose in this dissertation for the multi-depot routing problem. By using the max-flow/min-cut theorem, we derive an equivalent system of inequalities, in terms of the corresponding linear programming relaxation, which does not require the double network flow systems and, thus, it is a system of inequalities in the space of the x, the v and the z variables. This system of inequalities is essentially the 3I^{++} system with the addition of a set of exponentially-many constraints, which can be interpreted as cut constraints for each specific circuit. We also derive additional sets of constraints in the space of the x and the v variables based on this formulation, namely a new set which generalizes the constraints obtained by using concepts from the precedence constrained traveling salesman problem and which includes as a special case a different set of subtour elimination constraints.

Finally, in Section 4.4 we present separation algorithms, in Section 4.5 we present computational results to determine the linear programming relaxation values provided by some of the proposed constraints throughout the first two sections of this chapter and to compare them to the results presented in the previous chapter, and in Section 4.6 we finish with some concluding remarks.

4.2 Path elimination constraints based on client-depot assignment variables

In this section we present path elimination constraints based on the set of client-depot assignment variables, which we recall are defined as binary variables $v_d^i = 1$ if client $i \in C$ is assigned to, or equivalently is in the circuit of, depot $d \in D$, and $v_d^i = 0$ otherwise. For an easier reading of this section we advise using Figure 4.1 shown in Section 4.2.8 as a reference.

In Section 4.2.1, we start by defining a base compact system of inequalities in the space of the x and the v variables which prevents the existence of paths between different depots.

As we mentioned before, the client-depot assignment variables v are closely related to the precedence variables used in the precedence constrained (asymmetric) traveling salesman problem, since they can also be interpreted as indicating whether a given depot precedes a given client or not. In Section 4.2.2 we use this relationship to derive a stronger compact system of inequalities than the one defined in Section 4.2.1. Then, in Section 4.2.3, we generalize the system of inequalities presented in Section 4.2.2, again by using the relationship to the precedence constrained traveling salesman problem.

In Section 4.2.4 we introduce a generalization of the original system of inequalities presented in Section 4.2.1 which is motivated by the existence of multiple depots. In addition, we will observe in Section 4.2.5 that this generalization is unrelated to the one proposed in Section 4.2.3 and that, in fact, we can combine both generalizations and obtain a new system of

inequalities which generalizes the original system of inequalities of Section 4.2.1 in two ways simultaneously.

In Section 4.2.6 we show that the client-depot assignment variables v can be related to the arc-depot assignment variables z, originally presented in Section 2.3, and we use this relationship to derive other systems of inequalities. More precisely, we derive a new system of inequalities, which includes as a special case a new compact system of inequalities, and we show that it is theoretically different from the ones based on ideas from the precedence constrained traveling salesman problem. In Section 4.2.7 we generalize this system of inequalities.

In Section 4.2.8 we summarize the previous sections and, finally, in Section 4.2.9 we combine some of the constraints presented earlier in order to derive new sets of constraints in the space of the x variables.

4.2.1 A base model in the space of the x and the v variables

In this section we present a first system of inequalities in the space of the x and the v variables which models path elimination constraints. Intuitively, we want to establish a system of inequalities that ensures that the v variables are correctly defined and that the clients which are in the circuit of a given depot $d \in D$ are linked to one another, and to the depot d, and are not linked to the remaining depots nor to clients which are assigned to other depots. One way we can do that is as follows:

$$x_{di} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.1}$$

$$x_{id} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.2}$$

$$\sum_{d \in D} v_d^i = 1 \qquad \forall i \in C \tag{4.3}$$

$$v_d^i + x_{ij} \leq v_d^j + 1 \qquad \forall d \in D,\ \forall i, j \in C,\ i \neq j \tag{4.4}$$

$$v_d^i \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \tag{4.5}$$

Constraints (4.1) and (4.2) ensure that if an arc linking a depot $d \in D$ and a client $i \in C$, respectively in both directions, is used, then i must be in the circuit of depot d. Conversely, if i is not in the circuit of depot d then these two nodes cannot be linked. Constraints (4.3) guarantee that every client is in one and only one circuit. Finally, constraints (4.4) state that if a client $i \in C$ is in the circuit of a depot $d \in D$ and if the arc between i and another client $j \in C$ is used, then j must also be in the circuit of depot d. Conversely, if j is not in the circuit of depot d, then i cannot be simultaneously in the circuit of depot d and linked to j.

We will denote the system of inequalities (4.1)–(4.5) by CDA. This system of inequalities is sufficient to model the path elimination constraints of the multi-depot routing problem. In fact, consider an unfeasible path $(d_1, i_1, i_2, \ldots, i_k, d_2)$ in which $d_1, d_2 \in D,\ d_1 \neq d_2$ and

$i_1, i_2, \ldots, i_k \in C$. If $k = 1$, observe that from constraints (4.1)–(4.2) we can conclude that $x_{d_1,i_1} = x_{i_1,d_2} = 1$ which implies that both $v_{d_1}^{i_1} = 1$ and $v_{d_2}^{i_1} = 1$, which violates constraints (4.3). Suppose now that $k \geq 2$. In this case, if we use constraints (4.1) for d_1 and i_1, then $v_{d_1}^{i_1} = 1$. Then, if we use constraints (4.4) in succession for the arcs $(i_1, i_2), \ldots, (i_{k-1}, i_k)$, we obtain $v_{d_1}^{i_l} = 1,\ \forall l = 2, 3, \ldots, k$. Finally, if we use constraints (4.2) for d_2 and i_k we get $v_{d_2}^{i_k} = 1$. But this is impossible since we have both $v_{d_1}^{i_k} = 1$ and $v_{d_2}^{i_k} = 1$ which violates constraints (4.3). Observe that constraints (4.3) could be defined as less than or equal to constraints and we would still obtain a valid set of path elimination constraints. We will come back to this observation later on.

Another way in which we can see that the CDA system prevents unfeasible paths between two depots is as follows. Consider the unfeasible path $(d_1, i_1, i_2, \ldots, i_k, d_2)$ as defined above and the following constraints: (i) constraints (4.1) written for d_1 and i_1; (ii) constraints (4.4) written for the arcs $(i_1, i_2), \ldots, (i_{k-1}, i_k)$ and d_1; and (iii) constraints $v_{d_1}^{i_k} + x_{kd_2} \leq 1$, which are obtained by combining constraints (4.2) written for d_2 and i_k with constraints (4.3). For clarity, we have the following:

$$\begin{aligned} x_{d_1 i_1} &\leq v_{d_1}^{i_1} \\ v_{d_1}^{i_1} + x_{i_1 i_2} &\leq v_{d_1}^{i_2} + 1 \\ &\ldots \\ v_{d_1}^{i_{k-1}} + x_{i_{k-1} i_k} &\leq v_{d_1}^{i_k} + 1 \\ v_{d_1}^{i_k} + x_{i_k d_2} &\leq 1 \end{aligned}$$

By adding all the above constraints and observing that the terms on the v variables cancel out, we obtain a constraint in the space of the x variables as follows:

$$x_{d_1 i_1} + x_{i_1 i_2} + \ldots + x_{i_{k-1} i_k} + x_{i_k d_2} \leq k, \quad \forall d_1, d_2 \in D,\ d_1 \neq d_2,\ \forall \text{ distinct } i_1, \ldots, i_k \in C. \quad (4.6)$$

Constraints (4.6) are the most basic path elimination constraints in the space of the x variables that one could think of and they simply limit the number of arcs which can be used in an unfeasible path with $k + 1$ arcs to k. In practice, constraints (4.6) are hard to use since one constraint exists for every possible unfeasible path between two depots and it is not clear how to separate them apart from standard enumeration. Nevertheless, in theory, they are path elimination constraints and, as we have shown, they are implied by the CDA system. For simplification, we will not be repeating these arguments in the subsequent sections, except in Section 4.2.9, however, other constraints in the space of the x variables can be obtained by repeating the above construction.

Note that the following reversed version of constraints (4.4) is also valid

$$v_d^i + x_{ji} \leq v_d^j + 1 \qquad \forall d \in D,\ \forall i, j \in C,\ i \neq j, \quad (4.7)$$

for reasons similar to the case of constraints (4.4).

We can easily see that the CDA system in which we replace the original constraints (4.4) by their reversed ones (4.7) can also model the path elimination constraints. In addition, the linear programming relaxations of these two systems of inequalities are not equivalent and so it makes sense to consider a third system of inequalities which includes constraints (4.4) and (4.7) simultaneously. However, we show in the following section that any of these two cases is dominated by a stronger system of inequalities. In fact, based on the relationship between the client-depot assignment variables and the precedence variables of the precedence constrained traveling salesman problem, we will show that both constraints (4.4) and (4.7) can be strengthened.

4.2.2 Strengthening the base model

Observe that, if we consider two clients $i, j \in C,\ i \neq j$, we can never use an arc (i, j) and its reverse (j, i) simultaneously or else we would form a subtour with these two client nodes. Consider then the following constraints for which the validity is easy to establish and which dominate both constraints $v_d^i + x_{ij} \leq v_d^j + 1$ (4.4) and constraints $v_d^i + x_{ji} \leq v_d^j + 1$ (4.7):

$$v_d^i + x_{ij} + x_{ji} \leq v_d^j + 1 \qquad \forall d \in D,\ \forall i, j \in C,\ i \neq j. \tag{4.8}$$

Recall that the client-depot assignment variables can be interpreted as precedence variables and, in fact, these lifted constraints follow the same line of thought as the one leading to the so-called disaggregated Desrochers and Laporte (DDL) constraints proposed by Gouveia & Pires (1999) (see, also, Desrochers & Laporte 1991) for the precedence constrained traveling salesman problem. We can now define a new system of inequalities, which we will denote by CDA-DDL, as follows:

$$x_{di} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.1}$$

$$x_{id} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.2}$$

$$\sum_{d \in D} v_d^i = 1 \qquad \forall i \in C \tag{4.3}$$

$$v_d^i + x_{ij} + x_{ji} \leq v_d^j + 1 \qquad \forall d \in D,\ \forall i, j \in C,\ i \neq j \tag{4.8}$$

$$v_d^i \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \tag{4.5}$$

Note that the CDA-DDL system is also better in practice than the CDA system of Section 4.2.1 since it remains compact, it also models path elimination constraints, it has the same number of constraints and it provides a stronger linear programming relaxation value.

4.2.3 Generalizations based on arc subsets

We still follow the work for the precedence constrained traveling salesman problem in this section, namely the work by Gouveia & Pires (2001), to show that constraints $v_d^i + x_{ij} + x_{ji} \leq v_d^j + 1$ (4.8) can be generalized for arc sets defining a clique.

Consider a client subset $S \subset C$ with at least two nodes. Intuitively, if we use $|S| - 1$ arcs of the clique defined by S, then there exists a path which links all nodes of S, in particular the start and end nodes of the path, say i and j, respectively. Then, clearly, if i is in the circuit of a depot $d \in D$, then j must also be in the same circuit. Therefore, we can derive the following set of constraints:

$$v_d^i + x(S) \leq v_d^j + |S| - 1 \qquad \forall d \in D,\ \forall S \subset C:\ 2 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j. \quad (4.9)$$

These constraints include constraints (4.8) as a special case when $S = \{i, j\}$. A formal proof of their validity will be given further on, however, observe that they are only of interest when $x(S) = |S| - 1$, that is, only if there is a path linking all nodes of S, and, therefore, we can apply the intuition given above. We can thus define a new system of inequalities which models path elimination constraints as follows:

$$x_{di} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \quad (4.1)$$

$$x_{id} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \quad (4.2)$$

$$\sum_{d \in D} v_d^i = 1 \qquad \forall i \in C \quad (4.3)$$

$$v_d^i + x(S) \leq v_d^j + |S| - 1 \qquad \forall d \in D,\ \forall S \subset C:\ 2 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j \quad (4.9)$$

$$v_d^i \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \quad (4.5)$$

We denote this system of inequalities by CDA-GDDL. Notice that constraints (4.9) are in exponential number and, therefore, in order to use this system of inequalities in practice we require a separation algorithm for constraints (4.9), which was not necessary in the case of the CDA and the CDA-DDL systems of Sections 4.2.1 and 4.2.2, respectively. In Section 4.4.2 we provide a polynomial-time separation algorithm for constraints (4.9).

Observe also that if we add a constraint (4.9) for a pair $i, j \in S,\ i \neq j$ to the reversed constraint (4.9) in which the roles of i and j are swapped, we obtain a subtour elimination constraint $x(S) \leq |S| - 1$ (2.1) for the set S. Thus, the CDA-GDDL system models both path elimination constraints and subtour elimination constraints.

4.2.4 Generalizations based on depot subsets

In Section 4.2.1 we presented the CDA system that could model path elimination constraints. One of the characteristics of this system of inequalities is that its constraints $v_d^i + x_{ij} \leq v_d^j + 1$

(4.4) are only defined for single depots. In this section we show how to generalize constraints (4.4) based on extending their underlying idea to subsets of depots.

Recall that the basic idea of constraints (4.4) is that whenever a client $i \in C$ is in the circuit of a depot $d \in D$ and an arc (i, j) is used, then the client $j \in C$ must also be in the circuit of depot d. Suppose now that we consider a subset $D' \subset D$. Clearly, the same reasoning applies to this subset D', that is, if client i is in the circuit of one of the depots of D' and an arc (i, j) is used, then j must also be in the circuit of one of the depots of D'.

More formally, we can consider the following generalization of constraints (4.4), which is based on a similar concept developed by Erdoğan, Laporte & Rodríguez-Chía (2016) for the Hamiltonian p-median problem (which is the topic of the second part of this dissertation):

$$v^i_{D'} + x_{ij} \leq v^j_{D'} + 1 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j. \tag{4.10}$$

These constraints are valid since, from constraints $\sum_{d \in D} v^i_d = 1$ (4.3), we have $v^i_{D'} \leq 1$ and, thus, the same reasoning of the non-generalized version (4.4) applies. In addition, if $|D'| = 1$ we obtain constraints (4.4) as a special case. Observe that constraints (4.10) are exponentially many, due to being defined for all subsets of the set D, however, they can be separated in polynomial time as we will show in Section 4.4.1, once again based on an idea by Erdoğan et al. (2016).

By using constraints (4.10) we can now define a new system of inequalities which can model path elimination constraints, and which we denote by GCDA:

$$x_{di} \leq v^i_d \qquad \forall d \in D,\ \forall i \in C \tag{4.1}$$

$$x_{id} \leq v^i_d \qquad \forall d \in D,\ \forall i \in C \tag{4.2}$$

$$\sum_{d \in D} v^i_d = 1 \qquad \forall i \in C \tag{4.3}$$

$$v^i_{D'} + x_{ij} \leq v^j_{D'} + 1 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j \tag{4.10}$$

$$v^i_d \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \tag{4.5}$$

One might be inclined to assume that this system of inequalities and the CDA-GDDL system presented in the previous section are simply different ways of generalizing the base CDA system of Section 4.2.1. However, we can actually combine both generalization ideas to derive a new system of inequalities which dominates all other systems of inequalities in the space of the x and the v variables presented thus far, as we will see in the following section.

4.2.5 Generalizations based on arc subsets and depot subsets

The observation made at the beginning of Section 4.2.4 also applies to the CDA-DDL system of Section 4.2.2 and the CDA-GDDL system of Section 4.2.3. More precisely, both constraints $v^i_d + x_{ij} + x_{ji} \leq v^j_d + 1$ (4.8) of the CDA-DDL system and constraints $v^i_d + x(S) \leq v^j_d + |S| - 1$ (4.9)

of the CDA-GDDL system, are only defined for single depots. Interestingly, the generalization based on depot subsets presented in Section 4.2.4 also applies to these two sets of constraints.

Recall that the argument used to obtain constraints (4.9) from the original set of constraints $v_d^i + x_{ij} \leq v_d^j + 1$ (4.4) is that the reasoning for the validity of these constraints can be applied as long as the two nodes i and j are linked through a path, and not necessarily only through an arc. But then, the argument which we used in Section 4.2.4 to obtain the generalized constraints $v_{D'}^i + x_{ij} \leq v_{D'}^j + 1$ (4.10) from constraints (4.4), which we recall states that if i and j are linked then both nodes must be assigned to the same subset of depots, does not conflict with the previous argument. Thus, we can combine both arguments and obtain a set of constraints which generalizes constraints (4.4) for arc subsets and depot subsets simultaneously:

$$v_{D'}^i + x(S) \leq v_{D'}^j + |S| - 1 \qquad \forall D' \subset D,\ \forall S \subset C:\ 2 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j. \tag{4.11}$$

These constraints are also in exponential number, however, they can be separated in polynomial time as we will show in Section 4.4.2. Clearly, if $S = \{i, j\}$ we obtain a generalization of constraints (4.8). Note that, even if this case is included in constraints (4.11), it is interesting to consider it separately since we can use a dedicated separation algorithm whenever $|S| = 2$ which is much faster than the one used for the more general set (4.11) and which we will present in Section 4.4.1. For completeness, we will specifically write this case:

$$v_{D'}^i + x_{ij} + x_{ji} \leq v_{D'}^j + 1 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j \tag{4.12}$$

With these new sets of constraints we can define two new systems of inequalities which model path elimination constraints. We will denote by GCDA-DDL the following system:

$$x_{di} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.1}$$

$$x_{id} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.2}$$

$$\sum_{d \in D} v_d^i = 1 \qquad \forall i \in C \tag{4.3}$$

$$v_{D'}^i + x_{ij} + x_{ji} \leq v_{D'}^j + 1 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j \tag{4.12}$$

$$v_d^i \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \tag{4.5}$$

By replacing constraints (4.12) with their generalization (4.11) in the above system of inequalities, we create a different system of inequalities which we denote by GCDA-GDDL. Note that, since constraints (4.11) generalize constraints (4.9), then the GCDA-GDDL system also models subtour elimination constraints, given the observation at the end of Section 4.2.3.

Out of all the different systems of inequalities presented thus far in the space of the x and the v variables, the GCDA-GDDL is the most general one, however, most of the arguments used were motivated by the knowledge of studies on the precedence constrained traveling salesman

problem and on the Hamiltonian p-median problem. In the following section we derive new systems of inequalities which can be seen as a different way of strengthening and generalizing the base model of Section 4.2.1 and which are closely related to the arc-depot assignment variable based systems of inequalities presented in Section 2.3.

4.2.6 Exploring the relationship between the v and the z variables in order to strengthen the base model

In Section 2.3 we presented several systems of inequalities based on the arc-depot assignment variables z. For clarity, and in order to facilitate the comprehension of this text, we will rewrite the definition of these variables and of the 3I^{++} system, which was the strongest system of inequalities presented that uses the z variables.

The arc-depot assignment variables are binary variables defined as $z_{ij}^d = 1$ if arc $(i,j) \in A$ is used in the circuit of depot $d \in D$, and $z_{ij}^d = 0$ otherwise. Recall that an arc with an endpoint in a depot $d \in D$ cannot be used in the circuit of another depot and, thus, variables z_{ij}^d, for any $d \in D$, are only defined for arcs $(i,j) \in A^C \cup A_O^d \cup A_I^d$. Additionally, recall the argument that stated that any arc which is used must be used in exactly one circuit. Based on these variables and these arguments, we defined the 3I^{++} system which is as follows:

$$\sum_{j \in C} z_{dj}^d = 1 \qquad \forall d \in D \tag{2.3}$$

$$\sum_{j \in C} z_{jd}^d = 1 \qquad \forall d \in D \tag{2.4}$$

$$\sum_{j \in \{d\} \cup C} z_{ji}^d = \sum_{j \in \{d\} \cup C} z_{ij}^d \qquad \forall d \in D,\ \forall i \in C \tag{2.5}$$

$$\sum_{d \in D} z_{ij}^d = x_{ij} \qquad \forall (i,j) \in A^C \tag{2.10}$$

$$z_{ij}^d = x_{ij} \qquad \forall d \in D,\ \forall (i,j) \in A_O^d \cup A_I^d \tag{2.11}$$

$$z_{ij}^d \in \{0,1\} \qquad \forall d \in D,\ \forall (i,j) \in A^C \cup A_O^d \cup A_I^d. \tag{2.7}$$

In this section we theoretically compare the above modeling approach to the one based on the client-depot assignment variables v. More precisely, we show that the z and the v variables are closely related and we use that relationship to derive a system of inequalities in the space of the x and the v variables which is dominated by a system of inequalities in the space of the x, the v and the z variables which is equivalent in terms of the corresponding linear programming relaxation to the 3I^{++} system.

Given the definition of the v and the z variables, we can easily see that they can be related as follows. Observe that the indegree of a client $i \in C$ in the circuit of a depot $d \in D$ indicates whether i is in the circuit of d or not. In other words, if the indegree of client i in the circuit of

depot d is 1, then i is certainly in the circuit of depot d. Conversely, if the indegree of client i in the circuit of depot d is 0, then i is certainly not in the circuit of depot d. This can be expressed in the following way:

$$\sum_{j \in \{d\} \cup C} z_{ji}^{d} = v_{d}^{i} \qquad \forall d \in D,\ \forall i \in C. \tag{4.13}$$

Note that by adding these equalities to the 3I^{++} system, along with the domain constraints for the v variables (4.5), the linear programming relaxation value is not altered. Additionally, by using the flow conservation constraints (2.5) we can derive the following similar equalities

$$\sum_{j \in \{d\} \cup C} z_{ij}^{d} = v_{d}^{i} \qquad \forall d \in D,\ \forall i \in C, \tag{4.14}$$

which can be interpreted as equalities (4.13) but now with respect to the outdegree of a client $i \in C$ in the circuit of a depot $d \in D$. Clearly we only need to add either (4.13) or (4.14) to the 3I^{++} system under the presence of the flow conservation constraints (2.5), however, for simplification, we will use both equalities indiscriminately in the ensuing text.

The point of presenting equalities (4.13) and (4.14) is that we can use them in order to compare both modeling approaches. To demonstrate exactly how, we start by showing that the 3I^{++} system to which we add the relationships (4.13)–(4.14) and the domain constraints for the v variables (4.5) dominates the following system of inequalities, which we will denote by GCDA^{+}:

$$x_{di} \leq v_{d}^{i} \qquad \forall d \in D,\ \forall i \in C \tag{4.1}$$

$$x_{id} \leq v_{d}^{i} \qquad \forall d \in D,\ \forall i \in C \tag{4.2}$$

$$\sum_{d \in D} v_{d}^{i} = 1 \qquad \forall i \in C \tag{4.3}$$

$$v_{D'}^{i} + x_{ij} + x(D', j) + x(i, D \setminus D') \leq v_{D'}^{j} + 1 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j \tag{4.15}$$

$$v_{d}^{i} \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \tag{4.5}$$

In this system of inequalities, constraints $v_{d}^{i} + x_{ij} \leq v_{d}^{j} + 1$ (4.4) of the original CDA system of Section 4.2.1 were replaced by constraints (4.15), hence, the GCDA^{+} system dominates the original CDA system. In fact, we can easily see that it actually dominates the GCDA system (hence its name) presented in Section 4.2.4, since constraints (4.15) of the GCDA^{+} system dominate constraints $v_{D'}^{i} + x_{ij} \leq v_{D'}^{j} + 1$ (4.10).

Proposition 11. *The 3I^{++} system to which we add the relationships* (4.13)–(4.14) *and the domain constraints* (4.5) *dominates the GCDA^{+} system and, consequently, the GCDA and the CDA systems.*

Proof. To prove this result we must show that the 3I^{++} system to which we add the relationships (4.13)–(4.14) and the domain constraints (4.5) implies constraints (4.1)–(4.3) and (4.15).

Regarding constraints (4.1) and (4.2) notice that from the relationships (4.13)–(4.14) for nodes $d \in D$ and $i \in C$ and the linking constraints between the z and the x variables (2.11) we get:

$$v_d^i = \sum_{j \in \{d\} \cup C} z_{ji}^d \geq z_{di}^d = x_{di}, \text{ and } v_d^i = \sum_{j \in \{d\} \cup C} z_{ij}^d \geq z_{id}^d = x_{id}.$$

For constraints (4.3), consider $i \in C$ and notice that if we add the relationships (4.13) for all depots of D we get:

$$\sum_{d \in D} v_d^i = \sum_{d \in D} \sum_{j \in \{d\} \cup C} z_{ji}^d = \sum_{d \in D} z_{di}^d + \sum_{d \in D} \sum_{j \in C} z_{ji}^d.$$

Then, by using the linking constraints between the z and the x variables (2.10) and (2.11) and, subsequently, the client indegree constraints (1.5) we get:

$$\sum_{d \in D} z_{di}^d + \sum_{d \in D} \sum_{j \in C} z_{ji}^d = x(D, i) + x(C, i) = 1.$$

In the case of constraints (4.15), consider a subset $D' \subset D$ and a client $j \in C$. If we add the relationships (4.13) for the depots in D' we obtain:

$$v_{D'}^j = \sum_{d \in D'} \sum_{k \in \{d\} \cup C} z_{kj}^d = \sum_{d \in D'} z_{dj}^d + \sum_{d \in D'} \sum_{k \in C} z_{kj}^d \geq \sum_{d \in D'} z_{dj}^d + \sum_{d \in D'} z_{ij}^d,$$

for some $i \in C \setminus \{j\}$. Now, if we add $\sum_{d \in D'} \sum_{k \in \{d\} \cup C:\, k \neq j} z_{ik}^d$ to both sides and use the relationships (4.14) for client i added up for the depots in D' we obtain:

$$v_{D'}^j + \sum_{d \in D'} \sum_{k \in \{d\} \cup C:\, k \neq j} z_{ik}^d \geq \sum_{d \in D'} z_{dj}^d + \sum_{d \in D'} \sum_{k \in \{d\} \cup C} z_{ik}^d = \sum_{d \in D'} z_{dj}^d + v_{D'}^i.$$

Then, by using the linking constraints (2.10) and (2.11) we derive:

$$v_{D'}^j + x(i, D') + x(i, C \setminus \{j\}) \geq x(D', j) + v_{D'}^i.$$

Finally, if we use the client outdegree constraints (1.4) for client i we can rewrite the above inequality as

$$v_{D'}^j + 1 \geq x(D', j) + x(i, D \setminus D') + x_{ij} + v_{D'}^i,$$

which is precisely a constraint (4.15). □

Recall that we observed in Section 2.3.1 that constraints (4.3) can actually be written as less than or equal to constraints. In this situation we can use the $3I^+$ system instead of the $3I^{++}$ system in Proposition 11. In other words, the difference between having equality or inequality in constraints (4.3) is similar to the difference between the $3I^+$ and $3I^{++}$ systems established in Proposition 2, that is, for an objective function which only depends on the x variables, no difference in the linear programming relaxation value will be observed.

The new constraints (4.15) can be separated in polynomial time, as we will see in Section 4.4.1. Additionally, we can derive a new compact system of inequalities to model path elimination constraints by considering the special case of constraints (4.15) in which $|D'| = 1$, that is, the following constraints:

$$v_d^i + x_{ij} + x_{dj} + x(i, D \setminus \{d\}) \leq v_d^j + 1 \qquad \forall d \in D,\ \forall i, j \in C,\ i \neq j. \tag{4.16}$$

More precisely, the system of inequalities comprised of constraints (4.1)–(4.3), the above constraints (4.16) and the domain constraints (4.5), which we denote by CDA$^+$, is a compact system of inequalities which dominates the original CDA system.

Notice that it is not possible to further lift constraints (4.15) with the term x_{ji} like in the case of constraints $v_{D'}^i + x_{ij} + x_{ji} \leq v_{D'}^j + 1$ (4.12) since it is possible to have $x_{ji} = x(i, D \setminus D') = 1$ and $v_{D'}^j = 0$. Therefore, this new way of lifting the original constraints (4.4) is not comparable in theory and in terms of the corresponding linear programming relaxation to any of the lifted and/or generalized sets of constraints presented in Sections 4.2.2, 4.2.3 and 4.2.5, which were based on ideas from the precedence constrained traveling salesman problem.

In the following section we present an additional result with the purpose of showing that we can also use the relationships between the v and the z variables (4.13)–(4.14) to derive generalizations of constraint sets. As an example, we will show that constraints (4.15) can be generalized. Note that they are already generalized for depot subsets, in a similar way as what was presented in Section 4.2.4, however, they are not generalized for arc sets. As we mentioned above, it is not possible to generalize them to arc sets defining a clique, but we can show that the single node subsets $\{i\}$ and $\{j\}$ can be generalized to client subsets with more than one node.

4.2.7 Generalizations based on client subsets

In this section we show how to generalize the single node subsets $\{i\}$ and $\{j\}$ of constraints $v_{D'}^i + x_{ij} + x(D', j) + x(i, D \setminus D') \leq v_{D'}^j + 1$ (4.15) to client subsets with more than one node. We could have started from the more general case in the previous section, since the proof is an extension of the proof of Proposition 11, however, for clarity, we decided to separate both cases.

The following result shows that the 3I^{++} system, presented in Section 2.3.3, and recalled in the previous section, to which we add the relationships between the v and the z variables (4.13)–(4.14) and the domain constraints for the v variables (4.5) implies the following set of constraints:

$$v_{D'}^{S_1} + x(S_1, S_2) + x(D', S_2) + x(S_1, D \setminus D') \leq v_{D'}^{S_2} + |S_1| \qquad \forall D' \subset D,\ \forall S_1, S_2 \subset C,\ S_1 \cap S_2 = \emptyset. \tag{4.17}$$

Proposition 12. *The 3I^{++} system to which we add the relationships* (4.13)–(4.14) *and the domain constraints* (4.5) *implies constraints* (4.17)*.*

Proof. Let $D' \subset D$ and $S_1, S_2 \subset C,\ S_1 \cap S_2 = \emptyset$ and consider the following inequality

$$v_{D'}^{S_2} = \sum_{d \in D'} \sum_{i \in S_2} \sum_{j \in \{d\} \cup C} z_{ji}^d \geq \sum_{d \in D'} \sum_{i \in S_2} \sum_{j \in \{d\} \cup S_1} z_{ji}^d = \sum_{d \in D'} \sum_{i \in S_2} z_{di}^d + \sum_{d \in D'} \sum_{i \in S_2} \sum_{j \in S_1} z_{ji}^d,$$

which can be obtained by adding the relationships (4.13) for $d \in D'$ and for $i \in S_2$. Now, by adding the appropriate terms to both sides, namely $\sum_{d \in D'} \sum_{i \in \{d\} \cup C \setminus S_2} \sum_{j \in S_1} z_{ji}^d$, and then using the relationships (4.14) added for $d \in D'$ and $j \in S_1$ we obtain:

$$v_{D'}^{S_2} + \sum_{d \in D'} \sum_{i \in \{d\} \cup C \setminus S_2} \sum_{j \in S_1} z_{ji}^d \geq \sum_{d \in D'} \sum_{i \in S_2} z_{di}^d + \sum_{d \in D'} \sum_{i \in \{d\} \cup C} \sum_{j \in S_1} z_{ji}^d = \sum_{d \in D'} \sum_{i \in S_2} z_{di}^d + v_{D'}^{S_1}.$$

Then, by using the linking constraints between the z and the x variables (2.10)–(2.11) on the above inequality we can derive:

$$v_{D'}^{S_2} + x(S_1, D') + x(S_1, C \setminus S_2) \geq x(D', S_2) + v_{D'}^{S_1}.$$

Finally, by using the client outdegree constraints (1.4) added for all nodes of S_1, we can rewrite the above inequality as

$$v_{D'}^{S_2} + |S_1| \geq x(D', S_2) + x(S_1, D \setminus D') + x(S_1, S_2) + v_{D'}^{S_1},$$

which completes the proof. □

By generalizing the sets $\{i\}$ and $\{j\}$ of constraints (4.15) to client subsets with more than one node in constraints (4.17), we can define a new system of inequalities in the space of the x and the v variables which models path elimination constraints and which dominates the GCDA$^+$ system presented in the previous section. However, constraints (4.17) do not seem to be separable in polynomial time, hence, their use in practice is limited and we will not provide computational results to assess the linear programming relaxation values which we could obtain by using constraints (4.17).

We would like to end this section with an important observation. We clearly stated in Propositions 11 and 12 that we were adding the relationships (4.13)–(4.14) and the domain constraints (4.5) to the 3I^{++} system. This is absolutely required unless the v variables are not used in any other constraints other than the relationships (4.13)–(4.14) and the domain constraints (4.5), in which case using the 3I^{++} system would suffice.

4.2.8 Summary

In the previous sections we presented several systems of inequalities in the space of the x and the v variables to model path elimination constraints. Given the number of different systems of inequalities, how they were derived, and how they compare in theory, we believe it is important to present a summary. As a complement, in Figure 4.1 we provide a visual representation of this

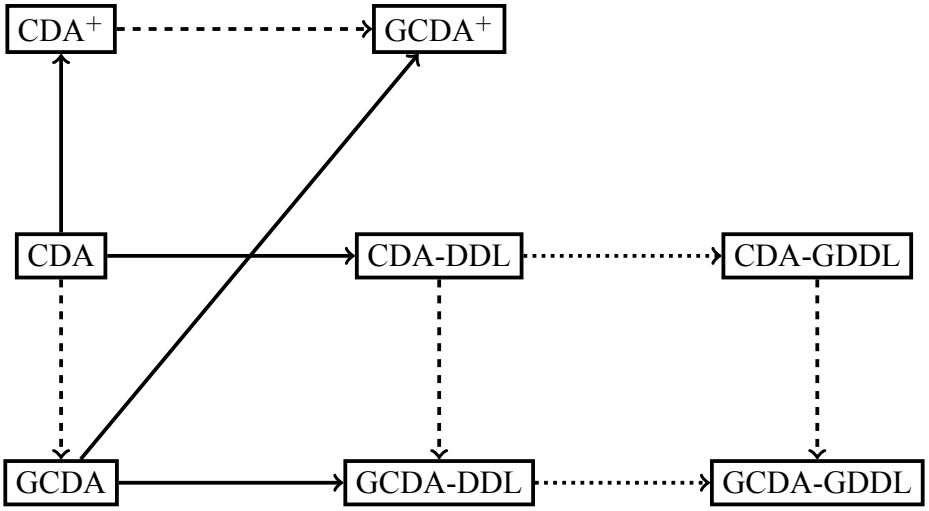

Figure 4.1: Summary of the systems of inequalities presented in Section 4.2

summary. A filled arrow indicates a lifting operation, a dashed arrow indicates a depot subset generalization and a dotted arrow indicates an arc subset generalization. Observe that these relationships are transitive and, thus, it is possible to easily establish a comparison of any two systems of inequalities in terms of their corresponding linear programming relaxation, if any exists. Recall that all of these systems of inequalities have a common set of constraints which are the following:

$$x_{di} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.1}$$

$$x_{id} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.2}$$

$$\sum_{d \in D} v_d^i = 1 \qquad \forall i \in C \tag{4.3}$$

$$v_d^i \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \tag{4.5}$$

The most basic system of inequalities, presented in Section 4.2.1 and denoted by CDA, uses constraints $v_d^i + x_{ij} \leq v_d^j + 1$ (4.4) in addition to the ones above. Each of the other systems of inequalities presented is obtained from this base system of inequalities by replacing constraints (4.4) with a different set which implies and/or generalizes the original constraints (4.4).

The first enhancements of the CDA system that we presented were derived from ideas similar to the ones used in the precedence constrained traveling salesman problem. In Section 4.2.2 we presented the CDA-DDL system that uses constraints $v_d^i + x_{ij} + x_{ji} \leq v_d^j + 1$ (4.8), which are based on the idea that arcs (i, j) and (j, i) cannot be simultaneously used for client nodes $i, j \in C,\ i \neq j$. By extending this idea to client subsets $S \subset C$ in Section 4.2.3, we derived constraints $v_d^i + x(S) \leq v_d^j + |S| - 1$ (4.9), which replace the original constraints (4.4) in the CDA-GDDL system.

In Section 4.2.4 we derived a different generalization of the original constraints (4.4) based on ideas similar to ones used in the Hamiltonian p-median problem. In particular, we presented

the GCDA system which uses constraints $v^i_{D'} + x_{ij} \leq v^j_{D'} + 1$ (4.10), where D' is a depot subset. Essentially, the argument used is that the intuition of the original constraints (4.4), which was applied to individual depots, can also be applied to depot subsets. This generalization is different from the one of the CDA-GDDL system, however, in Section 4.2.5 we showed that they can be combined and we derived two additional systems of inequalities, the GCDA-DDL system and the GCDA-GDDL system which use, respectively, constraints $v^i_{D'} + x_{ij} + x_{ji} \leq v^j_{D'} + 1$ (4.12) and $v^i_{D'} + x(S) \leq v^j_{D'} + |S| - 1$ (4.11).

Finally, in Section 4.2.6 we presented an additional two new systems of inequalities that enhance the original CDA system. These new systems of inequalities were derived from the relationship between the client-depot assignment variables v and the arc-depot assignment variables z. One of the systems of inequalities, denoted by CDA$^+$, uses the set of constraints $v^i_d + x_{ij} + x_{dj} + x(i, D \setminus \{d\}) \leq v^j_d + 1$ (4.16), whereas the other system of inequalities, denoted by GCDA$^+$, uses constraints $v^i_{D'} + x_{ij} + x(D', j) + x(i, D \setminus D') \leq v^j_{D'} + 1$ (4.15). Additionally, in Section 4.2.7, we showed that the CDA$^+$ and the GCDA$^+$ systems can be further generalized, however, we do not believe the resulting system of inequalities is usable in practice.

An important classification of the different systems of inequalities presented arises from the difference between compact and non-compact systems. The CDA, CDA-DDL and CDA$^+$ are compact systems of inequalities to model path elimination constraints. Conversely, the remaining systems, that is, the CDA-GDDL, the GCDA, the GCDA-DDL, the GCDA-GDDL and the GCDA$^+$ systems, are non-compact systems of inequalities. For the non-compact systems of inequalities we will provide separation algorithms in subsequent sections.

Another important aspect to consider is how the different systems compare in theory. Clearly the CDA system is the weakest one since all other systems either imply and/or generalize it. From the original CDA system there are two ways of strengthening it which are incomparable. In one way we use ideas from the precedence constrained traveling salesman problem and in the other we use the relationship between the v and the z variables. In the former case, we obtain the CDA-DDL system, which can be generalized to the CDA-GDDL system, and, in the latter, we obtain the CDA$^+$ system. Then, an unrelated generalization based on depot subsets based on ideas from the Hamiltonian p-median problem can be applied to any of the above systems which originates the GCDA, the GCDA-DDL, the GCDA-GDDL and the GCDA$^+$ systems.

The generalizations proposed in this section are straightforward given that they were derived from similar constraints in related problems, however, we will revisit the space of the x and the v variables in a subsequent section and, in particular, we will show that we can further generalize the GCDA-GDDL system.

4.2.9 Deriving inequalities in the space of the x variables

In this section we revisit the space of the arc variables x with the objective of illustrating how we can derive new inequalities in this space by appropriately combining inequalities in the space of the x and the v variables. The study conducted in this section is preliminary, in the sense that we do not fully explore the space of the x variables, and certainly not from a practical point of view. Nevertheless, our objective in this dissertation is to provide important tools and hints so as to continue this study in the future.

The main technique that we use to combine inequalities in the space of the x and the v variables is based on adding inequalities such that there are the exact same terms with v variables on both the left-hand side and the right-hand side. This technique is not new and has been used in the precedence constrained traveling salesman problem (see, e.g., Gouveia & Pires 1999, 2001, Gouveia & Pesneau 2006). We have already used this technique in Section 4.2.1 and in Section 4.2.3, in the latter case to show that the CDA-GDDL system also models subtour elimination constraints since it implies constraints $x(S) \leq |S| - 1$ (2.1). In this section we use it to combine constraints arising from two ideas. More precisely, we combine: (i) constraints based on the precedence constrained traveling salesman problem; with (ii) the new constraints based on the relationship between the v and the z variables.

Consider constraints $v^i_{D'} + x_{ij} + x_{ji} \leq v^j_{D'} + 1$ (4.12) for a given $D' \subset D$ and a pair of distinct clients i and j, and constraints $v^i_{D'} + x_{ij} + x(D', j) + x(i, D \setminus D') \leq v^j_{D'} + 1$ (4.15) for the same depot subset D' and for the same clients i and j but in which the roles of i and j are swapped when compared to the other constraint, that is, $v^i_{D'} + x_{ij} + x_{ji} \leq v^j_{D'} + 1$ and $v^j_{D'} + x_{ji} + x(D', i) + x(j, D \setminus D') \leq v^i_{D'} + 1$. Observe that if we add these two constraints, then the terms on the v variables are eliminated and we obtain the following set of constraints in the space of the x variables:

$$x_{ij} + 2x_{ji} + x(D', i) + x(j, D \setminus D') \leq 2 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j. \tag{4.18}$$

This particular set of constraints can be related to the directed chain-barring constraints presented in Section 2.5, namely the directed chain-barring constraints for client sets with two nodes, which we recall are as follows:

$$x(D', i) + x(i, D') + 3x_{ij} + 3x_{ji} + x(j, D \setminus D') + x(D \setminus D', j) \leq 4 \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j. \tag{2.18}$$

Notice that by adding two constraints (4.18) for a depot subset $D' \subset D$ and the same two distinct client nodes i and j by swapping their roles, we obtain exactly a constraint (2.18), which means that constraints (4.18) dominate constraints (2.18).

In the following result we formalize the derivation of constraints (4.18). More precisely, we show that by appropriately combining constraints in the space of the x and the v variables we

can derive a set of constraints in the space of the x variables which includes constraints (4.18) as a special case.

Proposition 13. *By appropriately combining constraints of the family* $v^i_{D'} + x(S) \leq v^j_{D'} + |S| - 1$ (4.11) *and constraints of the family* (4.15)*, we can derive the following set of constraints in the space of the* x *variables:*

$$x(S) + x(D', i) + x(j, D \setminus D') + x_{ji} \leq |S| \\ \forall D' \subset D,\ \forall S \subset C:\ 2 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j. \tag{4.19}$$

Proof. Consider a depot subset $D' \subset D$, a client subset $S \subset C$ and let i and j be two distinct nodes of S. Consider also constraints $v^i_{D'} + x(S) \leq v^j_{D'} + |S| - 1$ (4.11) and constraints (4.15), with the former written for D', S, i and j, and the latter written for D', j and i, that is, $v^i_{D'} + x(S) \leq v^j_{D'} + |S| - 1$ and $v^j_{D'} + x_{ji} + x(D', i) + x(j, D \setminus D') \leq v^i_{D'} + 1$. By adding these two constraints we obtain a constraint (4.19) for D', S, i and j. □

Even though we will not be testing these constraints in practice, we will provide, in Section 4.4.4, a polynomial-time separation algorithm for constraints (4.19) since they are in exponential number.

Observe that, for $S = \{i, j\}$, constraints (4.19) are exactly constraints (4.18). Additionally, constraints (4.19) when $|S| \geq 3$ are also similar to the directed chain-barring constraints for clients sets with at least three nodes, which we recall are as follows

$$x(D', i) + x(i, D') + 2x(S) + x(j, D \setminus D') + x(D \setminus D', j) \leq 2|S| - 1 \\ \forall D' \subset D,\ \forall S \subset C: 3 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j, \tag{2.17}$$

however, in this case, no dominance relationship can be established. In fact, this is clearer if we note that constraints (4.19) are a stronger version of constraints

$$x(D', i) + x(S) + x(j, D \setminus D') \leq |S| \qquad \forall D' \subset D,\ \forall S \subset C,\ \forall i, j \in S,\ i \neq j, \tag{2.19}$$

which are a weaker version of the k-MCC inequalities $x(D', S) + x(S) + x(S, D \setminus D') \leq |S|$ (2.16) written in their alternative form presented in Section 2.5.2.

The most interesting point of constraints (4.19) is that they are the first (and only) set of constraints in the space of the x variables presented in this dissertation which partially model both subtour elimination and path elimination constraints. More precisely, consider that we have an unfeasible path between two depots $(d_1, i, k_1, \ldots, k_l, j, d_2)$, where $d_1 \in D'$, $d_2 \in D \setminus D'$, and $S = \{i, j, k_1, \ldots, k_l\}$. Then, constraint (4.19) for D', S, i and j is violated since the left-hand side has a value of $|S| + 1$. Additionally, suppose that the nodes of S form a subtour in which the arc (j, i) is used. In this case, the left-hand side of constraint (4.19) for D', S, i and j is again $|S| + 1$ and, thus, it is violated.

Clearly, from a purely integer point of view, these constraints cannot be used as subtour elimination constraints and/or path elimination constraints for every case, since there are many subtours and/or unfeasible paths which they do not prevent. However, they can potentially be interesting to use as valid inequalities in formulations defined in the space of the x variables. In order to better understand what these constraints do different from the standard subtour elimination constraints $x(S) \leq |S| - 1$ (2.1) and the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13), consider a fractional circuit $(d_1, i, j, d_2, j, i, d_1)$ in which every arc has value $\frac{1}{2}$, with $d_1 \in D'$, $d_2 \in D \setminus D'$, and i and j distinct nodes of C. Notice first that $x_{ij} + x_{ji} = 1$, and so the subtour elimination constraint (2.1) written for $S = \{i, j\}$ is not violated. In addition, one can also easily see that this circuit satisfies every 1-MCC inequality (2.13) written for d_1 or d_2 and $S = \{i, j\}$. However, observe that $x_{ij} + 2x_{ji} + x(D', i) + x(j, D \setminus D') = \frac{1}{2} + 2 \times \frac{1}{2} + \frac{1}{2} + \frac{1}{2} > 2$ and so one of the constraints (4.18) is violated.

Constraints (4.19) are the extent of the study of this section. However, the ideas here shown, in particular constraints (4.19) and the proof of Proposition 13, can potentially provide some insight for future research. We believe that many of the constraints which can be derived using similar techniques to these are constraints which mix subtour elimination and path elimination constraints, much like constraints (4.19) and, thus, are an interesting future investigation.

4.3 A formulation in the space of the x, the v and the z variables

In this section we present a new formulation in the space of the original arc variables x, the client-depot assignment variables v presented in Section 4.2, and the arc-depot assignment variables z presented in Section 2.3. This new formulation models subtour elimination constraints and path elimination constraints simultaneously and provides linear programming relaxation values which are very close to the optimal value in practice, as we will show in Section 4.5.2.

We start by presenting in Section 4.3.1 an adaptation of the double multi-commodity network flow systems proposed by Wong (1980) for the traveling salesman problem. Intuitively, these double network flows ensure that there must be flow coming from a depot to a client and from the same client to the same depot provided that said client is in the circuit of that specific depot. We then show that by relating these double network flows to the arc variables x we are able to obtain a valid set of path elimination constraints.

Most importantly, however, we show in Section 4.3.2 that if we relate these double network flow models to the arc-depot assignment variables z we obtain a formulation which models both subtour elimination constraints and path elimination constraints and, in Section 4.3.3, by using the max-flow/min-cut theorem to eliminate the double network flow systems, we derive an equivalent formulation, in terms of the corresponding linear programming relaxation, in the

space of the x, the v and the z variables which is more tractable in practice.

Finally, in Section 4.3.4 we use the formulation in the space of the x, the v and the z variables to derive additional constraints in the space of the x and the v variables.

4.3.1 Path elimination constraints based on double multi-commodity network flow systems

In this section we revisit the double (multi-commodity) network flow models proposed by Wong (1980) for the traveling salesman problem and adapt them to the multi-depot routing problem, in particular to the modeling of path elimination constraints. More precisely, we present an adaptation of the double network flow models to the multi-depot routing problem case in addition to sets of linking constraints that relate these flow models to the original arc variables x, which allows us to derive a system of inequalities which prevents the existence of paths between different depots.

For these double network flow models we use the client-depot assignment variables v_d^i presented in Section 4.2, which we recall state whether or not a client $i \in C$ is in the circuit of a depot $d \in D$. Note that the v variables are auxiliary variables, that is, they can be eliminated from the constraints that we will present, however, we include them since it is much easier to explain the modeling approach. The double network flow models for the multi-depot routing problem are based on the following observation. If a client i is in the circuit of a depot d then there must exist a path from d to i that does not go through other depots, and there must exist a path from i to d that does not go through other depots. Equivalently, we can interpret the v_d^i variables as stating whether or not flow must be sent from d to i and, thus, v_d^i units of flow must go from d to i and cannot flow through other depots, and v_d^i units of flow must go from i to d and cannot flow through other depots.

We define two new sets of binary flow variables. Consider the binary variables $f_{pq}^{di} = 1$ if flow is sent from $d \in D$ to $i \in C$ via arc $(p,q) \in \left(A^C \setminus \{(i,j) : j \in C\}\right) \cup A_O^d$, and $f_{pq}^{di} = 0$ otherwise, and the binary variables $g_{pq}^{di} = 1$ if flow is sent from $i \in C$ to $d \in D$ via arc $(p,q) \in \left(A^C \setminus \{(j,i) : j \in C\}\right) \cup A_I^d$, and $g_{pq}^{di} = 0$ otherwise. Intuitively, the f flow models the flow coming from d and headed to i, whereas the g flow models the reserve. Consider, then, the following flow system for the f variables

$$\sum_{q \in C} f_{dq}^{di} = v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.20}$$

$$\sum_{p \in \{d\} \cup C} f_{pi}^{di} = v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.21}$$

$$\sum_{q \in \{d\} \cup C \setminus \{i\}} f_{qp}^{di} = \sum_{q \in C} f_{pq}^{di} \quad \forall d \in D,\ \forall i \in C,\ \forall p \in C \setminus \{i\} \tag{4.22}$$

$$f_{pq}^{di} \in \{0,1\} \qquad \forall d \in D,\ \forall i \in C,\ \forall (p,q) \in \left(A^C \setminus \{(i,j) : j \in C\}\right) \cup A_O^d, \tag{4.23}$$

and the following flow system for the g variables

$$\sum_{p \in C} g_{pd}^{di} = v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.24}$$

$$\sum_{q \in \{d\} \cup C} g_{iq}^{di} = v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.25}$$

$$\sum_{q \in C} g_{qp}^{di} = \sum_{q \in \{d\} \cup C \setminus \{i\}} g_{pq}^{di} \quad \forall d \in D,\ \forall i \in C,\ \forall p \in C \setminus \{i\} \tag{4.26}$$

$$g_{pq}^{di} \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C,\ \forall (p, q) \in \left(A^C \setminus \{(j, i) : j \in C\}\right) \cup A_I^d. \tag{4.27}$$

Regarding the f flow system, and for each pair (d, i) such that $d \in D$ and $i \in C$, constraints (4.20) and (4.21) ensure that the amount of flow leaving depot d and entering client i, respectively, is the same and it is equal to v_d^i, that is, if i is in the circuit of d, then 1 unit of flow leaves d with destination i, and if i is not in the circuit of d, then no f flow exists for the pair (d, i). Additionally, constraints (4.22) are the flow conservation constraints for the remaining nodes in $C \setminus \{i\}$ and, thus, they state that the amount of f flow for the pair (d, i) which enters a given client $p \in C \setminus \{i\}$ is equal to the flow which leaves said client. As for the g flow system the only difference when compared to the f flow system is that the g flow goes from i to d.

In order to obtain a valid set of path elimination constraints we require additional linking constraints between the f and g and the x variables as well as additional constraints involving the v variables, namely constraints $\sum_{d \in D} v_d^i = 1$ (4.3) and the domain constraints (4.5). Regarding the former, consider the following linking constraints for each arc $(d, q) \in A_O^d$

$$f_{dq}^{dq} = x_{dq} \qquad \forall (d, q) \in A_O^d \tag{4.28}$$

$$f_{dq}^{dk} \leq x_{dq} \qquad \forall (d, q) \in A_O^d,\ \forall k \in C \setminus \{q\}, \tag{4.29}$$

the following linking constraints for each arc $(p, d) \in A_I^d$

$$g_{pd}^{dp} = x_{pd} \qquad \forall (p, d) \in A_I^d \tag{4.30}$$

$$g_{pd}^{dk} \leq x_{pd} \qquad \forall (p, d) \in A_I^d,\ \forall k \in C \setminus \{p\}, \tag{4.31}$$

and, finally, the following linking constraints for each arc $(p, q) \in A^C$

$$\sum_{d \in D} f_{pq}^{dq} = x_{pq} \qquad \forall (p, q) \in A^C \tag{4.32}$$

$$\sum_{d \in D} g_{pq}^{dp} = x_{pq} \qquad \forall (p, q) \in A^C \tag{4.33}$$

$$\sum_{d \in D} \left(f_{pq}^{dk} + g_{pq}^{dk}\right) \leq x_{pq} \qquad \forall (p, q) \in A^C,\ \forall k \in C \setminus \{p, q\}. \tag{4.34}$$

Constraints (4.28) state that an arc $(d, q) \in A_O^d$ is used if and only if flow from d to q is sent via that arc. Constraints (4.29) ensure that if an arc $(d, q) \in A_O^d$ is used to send flow from d to a

node $k \in C \setminus \{q\}$, then the corresponding variable x_{dq} must be equal to 1. Conversely, if an arc $(d, q) \in A_O^d$ is not used then it cannot be used to send flow to k. Constraints (4.30)–(4.31) are similar to constraints (4.28)–(4.29) but with respect to arcs in A_I^d. Constraints (4.32) and (4.33) state that an arc $(p, q) \in A^C$ is used if and only if it is used to send flow from a depot $d \in D$ to q and if and only if it is used to send flow from p to a depot $d \in D$, respectively. Finally, constraints (4.34) ensure that if an arc $(p, q) \in A^C$ is used to send flow from any depot to a client $k \in C \setminus \{p, q\}$ or from k to any depot, then the corresponding variable x_{pq} must be equal to 1. Conversely, if the arc $(p, q) \in A^C$ is not used then it cannot be used to send any flow destined to or coming from k. Note that constraints (4.34) are valid since if we fix an arc $(p, q) \in A^C$ and a client $k \in C \setminus \{p, q\}$ then we know that the flow coming from any depot d to k uses a set of arcs that is disjoint from the set of arcs that is used to send flow from k back to d, that is, an arc is either used to send flow to k or to send flow from k.

The system of inequalities (4.20)–(4.34) to which we add constraints (4.3) and (4.5), which we denote by FG-X, defines a valid set of path elimination constraints. Intuitively, observe that an unfeasible path $(d_1, i_1, i_2, \ldots, i_k, d_2)$, in which $d_1, d_2 \in D,\ d_1 \neq d_2$ and $i_1, i_2, \ldots, i_k \in C$, would force any client node in it to be simultaneously in the circuit of depot d_1 and d_2 which would violate constraints (4.3). Note that both flow systems f and g are required to be in the FG-X system in order to obtain a valid set of path elimination constraints.

We will not be exploring the FG-X system of inequalities more than up to this point in this dissertation. Essentially, our objective in this section was only to present the double network flow models f and g and show that they can theoretically be used to model path elimination constraints and, thus, we will not compare this system to previous sets of path elimination constraints neither in theory nor in practice, the latter due to the fact that it is computationally impractical to use the FG-X system given its large number of both variables and constraints. Furthermore, the FG-X system is not as interesting in theory as the stronger system that we will present in the next section which is based on relating the double network flow models f and g to the arc-depot assignment variables z of Section 2.3.

4.3.2 Combining the systems of inequalities based on the z variables with the f and g flow systems

In the previous section we presented a set of double network flow systems, the f and g flow systems, which state that, essentially, for each pair (d, i) such that $d \in D$ and $i \in C$, flow must be sent from d to i and from i to d, respectively, provided that i is in the circuit of depot d. These flow systems were linked via a set of linking constraints with the arc variables x, however, we show in this section that we can derive a stronger system of inequalities if we establish linking constraints between the f and g variables and the arc-depot assignment variables z, originally

presented in Section 2.3. In particular, we are interested in combining the double network flow systems with the 3I^{++} system, which was presented in Section 2.3.3 and which, for clarity, we rewrite here:

$$\sum_{j \in C} z^d_{dj} = 1 \qquad \forall d \in D \tag{2.3}$$

$$\sum_{j \in C} z^d_{jd} = 1 \qquad \forall d \in D \tag{2.4}$$

$$\sum_{j \in \{d\} \cup C} z^d_{ji} = \sum_{j \in \{d\} \cup C} z^d_{ij} \qquad \forall d \in D,\ \forall i \in C \tag{2.5}$$

$$\sum_{d \in D} z^d_{ij} = x_{ij} \qquad \forall (i,j) \in A^C \tag{2.10}$$

$$z^d_{ij} = x_{ij} \qquad \forall d \in D,\ \forall (i,j) \in A^d_O \cup A^d_I \tag{2.11}$$

$$z^d_{ij} \in \{0,1\} \qquad \forall d \in D,\ \forall (i,j) \in A^C \cup A^d_O \cup A^d_I. \tag{2.7}$$

Intuitively, for arcs in A^C, notice that the z and the x variables are linked with linking constraints of the form $\sum_D z \leq x$ (2.10) in the 3I^{++} system, whereas the f and g flow systems were linked with the x variables with constraints of the form $\sum_D (f + g) \leq x$ (4.32)–(4.34). However, we will show that it is possible to define linking constraints between the f and g and the z variables, for the same arcs in A^C, which are of the form $f + g \leq z$, for any $d \in D$, and, thus, which are stronger than the original linking constraints between the f and g and the x variables.

More formally, consider an arc $(p,q) \in A^C$ and observe that if for some depot $d \in D$ flow from d to some $i \in C$ (or from i to d) flows on arc (p,q) then surely this arc must be used in the circuit of depot d. Conversely, if the arc (p,q) is not used in the circuit of depot d, then no flow from d destined to i can flow on that arc. In general, we can define linking constraints between the f and g and the z variables as follows. Consider the following linking constraints for each arc $(d,q) \in A^d_O$

$$f^{dq}_{dq} = z^d_{dq} \qquad \forall (d,q) \in A^d_O \tag{4.35}$$

$$f^{dk}_{dq} \leq z^d_{dq} \qquad \forall (d,q) \in A^d_O,\ \forall k \in C \setminus \{q\}, \tag{4.36}$$

the following linking constraints for each arc $(p,d) \in A^d_I$

$$g^{dp}_{pd} = z^d_{pd} \qquad \forall (p,d) \in A^d_I \tag{4.37}$$

$$g^{dk}_{pd} \leq z^d_{pd} \qquad \forall (p,d) \in A^d_I,\ \forall k \in C \setminus \{p\}, \tag{4.38}$$

and the following linking constraints for each arc $(p,q) \in A^C$

$$f^{dq}_{pq} = z^d_{pq} \qquad \forall (p,q) \in A^C,\ \forall d \in D \tag{4.39}$$

$$g^{dp}_{pq} = z^d_{pq} \qquad \forall (p,q) \in A^C,\ \forall d \in D \tag{4.40}$$

$$f_{pq}^{dk} + g_{pq}^{dk} \leq z_{pq}^{d} \qquad \forall (p,q) \in A^C,\ \forall d \in D,\ \forall k \in C \setminus \{p,q\}. \tag{4.41}$$

The interpretation of the linking constraints for arcs in A_O^d and A_I^d (4.35)–(4.38) is similar to the interpretation of the linking constraints between the f and g and the x variables (4.28)–(4.31) for the same sets of arcs. In addition, the interpretation of the linking constraints for arcs in A^C (4.39)–(4.41) was intuitively explained with the observation made prior.

Under the linking constraints (4.35)–(4.41), we can combine the 3I^{++} system (2.3)–(2.5), (2.10)–(2.11) and (2.7), and the two flow systems f (4.20)–(4.23) and g (4.24)–(4.27), presented in the previous section, with the addition of the relationships between the v and the z variables (4.13)–(4.14) and the domain constraints for the v variables (4.5). Note that constraints $\sum_{d \in D} v_d^i = 1$ (4.3) are no longer necessary since we proved in Proposition 11 that they are implied by the 3I^{++} system to which we add the relationships between the v and the z variables (4.13)–(4.14). The resulting system, which we denote by FG-Z, is clearly stronger than the FG-X system presented in the previous section. In addition, the FG-Z system models both path elimination constraints and subtour elimination constraints. The former is straightforward and the latter will be proved in the next section.

Observe, however, that the FG-Z system resulting from combining the $3I^{++}$ system with the double network flows f and g is even more impractical to use computationally when compared to the FG-X system presented in the previous section, specially given that both the number of variables and the number of constraints increased. In order to obtain computational results that allow us to study the linear programming relaxation values provided by the FG-Z system, we derived an equivalent system of inequalities in the space of the x, the v and the z variables which is more tractable in practice but is no longer a compact system of inequalities. This is the topic of the next section.

4.3.3 Eliminating the f and g flow systems by using the max-flow/min-cut theorem

In this section we show how to derive a system of inequalities in the space of the x, the v and the z variables which is equivalent, in terms of the corresponding linear programming relaxation, to the FG-Z system defined in the previous section. We will do so in two steps. Firstly, we prove that in the FG-Z system only one of the two network flow models is required, or, more precisely, one of the two network flow models is redundant in the corresponding linear programming relaxation. Recall that this was not true in the case of the FG-X system presented in Section 4.3.1, which linked the f and g variables with the x variables, and, in this case, it will allow us to considerably reduce the number of variables and constraints in the FG-Z system by completely discarding one of the flow models. Secondly, we eliminate the other remaining flow system by using the max-flow/min-cut theorem, thus effectively deriving a system of inequalities which

only uses the x, the v and the z variables and which is equivalent, in terms of the corresponding linear programming relaxation, to the FG-Z system. We start with the first result, which is similar to a result proved by Langevin, Soumis & Desrosiers (1990) in the context of the traveling salesman problem.

Proposition 14. *In the FG-Z system, comprised of the 3I^{++} system* (2.3)–(2.5), (2.10)–(2.11) *and* (2.7)*, the f flow system* (4.20)–(4.23)*, the g flow system* (4.24)–(4.27)*, the linking constraints between the f and g and the z variables* (4.35)–(4.41)*, the relationships between the v and the z variables* (4.13)–(4.14)*, and the domain constraints for the v variables* (4.5)*, one of the two flow systems f or g is redundant in the corresponding linear programming relaxation.*

Proof. We start by observing that certain z, f and g variables were not defined for some arcs of A, however, for simplification of this proof, we define all variables for all arcs by setting their value to 0 when necessary.

Given feasible flows z' and f', and a feasible auxiliary solution v', we construct a flow g' as follows:

$$g'^{di}_{pq} = z'^{d}_{pq} - f'^{di}_{pq} \qquad \forall d \in D,\ \forall i \in C,\ \forall (p,q) \in A. \tag{4.42}$$

Immediately we can see that all linking constraints between the f and g and the z variables are satisfied by the z', f' and g' flow systems.

Consider now a pair (d,i) such that $d \in D$ and $i \in C$. In order to complete the proof, we only need to verify that g' satisfies flow conservation constraints for the nodes d and i and all nodes $k \in C \setminus \{i\}$, which correspond to constraints (4.24)–(4.26) respectively.

Regarding the flow conservation constraints of depot d, observe first that, for the z' and the f' flows, we have:

$$\sum_{q \in V} z'^{d}_{dq} - \sum_{p \in V} z'^{d}_{pd} = 1 - 1, \qquad \text{and} \qquad \sum_{q \in V} f'^{di}_{dq} - \sum_{p \in V} f'^{di}_{pd} = v'^{i}_{d} - 0.$$

Therefore, for flow g' we can derive:

$$0 - \sum_{p \in V} g'^{di}_{pd} = \sum_{q \in V} g'^{di}_{dq} - \sum_{p \in V} g'^{di}_{pd} = 1 - 1 - (v'^{i}_{d} - 0) \Leftrightarrow \sum_{p \in V} g'^{di}_{pd} = v'^{i}_{d}.$$

With respect to node i, observe that

$$\sum_{q \in V} z'^{d}_{iq} - \sum_{p \in V} z'^{d}_{pi} = 0, \qquad \text{and} \qquad \sum_{q \in V} f'^{di}_{iq} - \sum_{p \in V} f'^{di}_{pi} = 0 - v'^{i}_{d},$$

and, thus, we have

$$\sum_{q \in V} g'^{di}_{iq} - 0 = \sum_{q \in V} g'^{di}_{iq} - \sum_{p \in V} g'^{di}_{pi} = 0 - (0 - v'^{i}_{d}) \Leftrightarrow \sum_{q \in V} g'^{di}_{iq} = v'^{i}_{d}.$$

Finally, for some $k \in C \setminus \{i\}$ we have

$$\sum_{q \in V} z'^{d}_{kq} - \sum_{p \in V} z'^{d}_{pk} = 0, \qquad \text{and} \qquad \sum_{q \in V} f'^{di}_{kq} - \sum_{p \in V} f'^{di}_{pk} = 0,$$

which implies that

$$\sum_{q \in V} g'^{di}_{kq} - \sum_{p \in V} g'^{di}_{pk} = 0.$$

We have therefore proved that g' is a feasible flow and, since it can be constructed from the z' and f' flows, it is redundant. Note that if we start from feasible flows z' and g', we can similarly derive a feasible flow f'. □

The result of Proposition 14 is important since it allows us to eliminate one of the two flow systems f or g from the FG-Z system and still ensure that the linear programming relaxation value is the same. Without loss of generality, assume that the g flow system is eliminated, that is, consider the system comprised of the 3I^{++} system (2.3)–(2.5), (2.10)–(2.11) and (2.7), the f flow system (4.20)–(4.23), the linking constraints between the f and the z variables (4.35), (4.36), (4.39) and (4.41) without the terms with the g variables, the relationships between the v and the z variables (4.13)–(4.14), and the domain constraints for the v variables (4.5), which we denote by F-Z system.

The linear programming relaxation of the F-Z system is equivalent to the linear programming relaxation of the FG-Z system and has fewer variables and constraints, however, it is still a complex system of inequalities which is hard to use in practice due to still including the f flow system. In the following result we show that we can derive a system of inequalities in the space of the x, the v and the z variables which is equivalent, in terms of the corresponding linear programming relaxation, to the F-Z system, and which uses a new set of exponentially-many constraints, by using the max-flow/min-cut theorem.

Proposition 15. *The projection of the linear programming relaxation of the f flow system, defined by* (4.20)–(4.23)*,* (4.35)–(4.36)*,* (4.39) *and* (4.41) *without the terms with the g variables, onto the space of the x, the v and the z variables is given by constraints $x_{ij} \geq 0,\ \forall (i,j) \in A$, $z^d_{ij} \geq 0,\ \forall d \in D,\ \forall (i,j) \in A^C \cup A^d_O \cup A^d_I$, and $v^i_d \geq 0,\ \forall d \in D,\ \forall i \in C$, and the following set of constraints:*

$$z^d(\{d\} \cup S', S) \geq v^i_d \qquad \forall d \in D,\ \forall S \subset C :\ 2 \leq |S| \leq |C| - |D|,\ \forall i \in S. \tag{4.43}$$

Proof. For each depot $d \in D$, consider a directed graph H_d with node set $\{d\} \cup C$ and arc set $A^d_O \cup A^C$. Given a fixed depot $d \in D$, the f flow system can be interpreted as a network flow system that guarantees that v^i_d units of flow are sent from d to every $i \in C$ in graph H_d with capacities given by the z variables. The max-flow/min-cut theorem indicates that v^i_d units of flow are sent from d to each node $i \in C$ if and only if every cut separating d from i has capacity

at least v_d^i. This last constraint is $z^d(\{d\} \cup S', S \cup \{i\}) \geq v_d^i$ for any $i \in C$ and any $S \subset C \setminus \{i\}$, which is an equivalent way of writing constraints (4.43). □

The main implication of Proposition 15 is that we can derive a new non-compact system of inequalities which is equivalent, in terms of the corresponding linear programming relaxation, to the F-Z system (and, consequently, to the original FG-Z system) but which no longer requires the f variables and, thus, is more manageable in practice provided we can efficiently separate constraints (4.43). In fact, given that the proof of Proposition 15 relies on the max-flow/min-cut theorem, we will be able to derive a polynomial-time separation algorithm for these constraints, which is presented in Section 4.4.3. We denote this new system of inequalities in the space of the x, the v and the z variables by ZV-CUTS, that is, the system comprised of the 3I^{++} system (2.3)–(2.5), (2.10)–(2.11) and (2.7), the new constraints (4.43), the relationships between the v and the z variables (4.13)–(4.14), and the domain constraints for the v variables (4.5).

An important property of the ZV-CUTS system is that it models both subtour elimination constraints and path elimination constraints. Regarding the latter, recall that the 3I^{++} system models path elimination constraints, hence, so does the ZV-CUTS system. As for the former, if we fix some subset $S \subset C$ and some $i \in S$, we add up constraints (4.43) for all depots $d \in D$, and we use constraints $\sum_{d \in D} v_d^i = 1$ (4.3) which are implied by the ZV-CUTS system, we obtain a subtour elimination constraint in cut form $x(D \cup S', S) \geq 1$ (2.2) for the set S, which means that the ZV-CUTS system also models subtour elimination constraints. Thus, by using the ZV-CUTS system to model both the generic subtour elimination constraints (1.6) and the generic path elimination constraints (1.7) of the generic formulation presented in Section 1.3, we obtain a valid formulation for the multi-depot routing problem. As we will see in Section 4.5.2, the formulation using the ZV-CUTS system provides linear programming relaxation values which are close to the optimal value in the instances tested.

Since the ZV-CUTS system is defined in the space of the x, the v and the z variables, an interesting investigation is to see which constraints we can derive in the space of the x and the v variables based on this system, similarly to what we did at the end of Section 4.2 going from the space of the x and the v variables to the space of the x variables.

4.3.4 Deriving inequalities in the space of the x and the v variables

In this section we present inequalities in the space of the x and the v variables implied by the ZV-CUTS system of Section 4.3.3. In particular, we show that the ZV-CUTS system implies a very general set of constraints, which includes as special cases both new constraints as well as constraints presented previously in Section 4.2. More formally, we start by showing that the ZV-CUTS system implies the following set of constraints:

$$x(D' \cup S', S) \geq v_{D_1}^{i_1} + \ldots + v_{D_k}^{i_k}$$

$$\forall \text{ partitions } D_1, \ldots, D_k \text{ of } D' \subseteq D,$$
$$\forall S \subset C: \ \max\{2, k\} \leq |S| \leq |C| - |D|, \ \forall \{i_1, \ldots, i_k\} \subseteq S. \tag{4.44}$$

Proposition 16. *The ZV-CUTS system, defined by the 3I^{++} system* (2.3)–(2.5)*,* (2.10)–(2.11) *and* (2.7)*, constraints* $z^d(\{d\} \cup S', S) \geq v_d^i$ (4.43)*, the relationships between the* v *and the* z *variables* (4.13)–(4.14)*, and the domain constraints for the* v *variables* (4.5)*, implies constraints* (4.44) *which are, therefore, valid for the multi-depot routing problem.*

Proof. Let $k \geq 1$ and consider $i_1, \ldots, i_k$ distinct client nodes of a set $S \subset C$ and $D_1, \ldots, D_k$ a partition of a subset of depots $D' \subseteq D$. For each $m = 1, \ldots, k$ consider the following constraints, which are of the family of constraints (4.43):

$$z^d(\{d\} \cup S', S) \geq v_d^{i_m} \qquad \forall d \in D_m$$

By adding all of the above constraints for $m = 1, \ldots, k$ and by using the linking constraints between the z and the x variables (2.10)–(2.11) of the 3I^{++} system, we obtain constraints (4.44). □

Constraints (4.44) include a number of interesting sets of constraints as special cases, as we will see. We conjecture that the exact separation of these constraints cannot be done in polynomial time, which means that, in practice, it is difficult to compute the linear programming relaxation values given by constraints (4.44). For this reason, we will focus on identifying two particular cases. Observe that, by appropriately using the client indegree constraints (1.5) for the nodes in set S, we can equivalently write constraints (4.44) as:

$$x(S) + x(D \setminus D', S) + v_{D_1}^{i_1} + \ldots + v_{D_k}^{i_k} \leq |S|$$
$$\forall \text{ partitions } D_1, \ldots, D_k \text{ of } D' \subseteq D,$$
$$\forall S \subset C: \ \max\{2, k\} \leq |S| \leq |C| - |D|, \ \forall \{i_1, \ldots, i_k\} \subseteq S. \tag{4.45}$$

In the above form we can immediately see that for $D' = D$ and $k = 2$, and by using constraints $\sum_{d \in D} v_d^i = 1$ (4.3), constraints (4.45) are constraints $v_{D'}^i + x(S) \leq v_{D'}^j + |S| - 1$ (4.11), which we recall were presented in Section 4.2.5 and which we had not yet proved to be valid. This observation, along with the result of Proposition 11, shows that the ZV-CUTS system dominates all systems of inequalities presented in Section 4.2. Another special case of interest is the case in which we set $k = 1$ in constraints (4.44). If $D' = D$ we obtain the regular subtour elimination constraints written in cut form $x(D \cup S', S) \geq 1$ (2.2). In the case in which D' is a proper subset of D we obtain the following constraints:

$$x(D' \cup S', S) \geq v_{D'}^i \qquad \forall D' \subset D, \ \forall S \subset C: \ 2 \leq |S| \leq |C| - |D|, \ \forall i \in S. \tag{4.46}$$

Constraints (4.46) are a different set of subtour elimination constraints, which are similar to sets of constraints used in formulations for the Hamiltonian p-median problem, namely in the

studies by Gollowitzer, Gouveia, Laporte, Pereira & Wojciechowski (2014) and Erdoğan et al. (2016).

Similarly to the case of Section 4.2.9, the purpose of this section is not to provide a complete study of inequalities in the space of the x and the v variables which we can derive from the ZV-CUTS system, but more so to provide tools which can be used in future research. For example, recall that, in Sections 4.2.6 and 4.2.7, we have already presented some results that show how to derive additional inequalities in the space of the x and the v variables by using the relationship between the v and the z variables and based on the 3I^{++} system, which is included in the ZV-CUTS system. In this section, we presented constraints (4.44) which, clearly, include many more particular cases which remain to be studied in more detail in future investigations.

4.4 Separation algorithms

In this section we present separation algorithms for some of the exponentially-sized sets of constraints presented throughout Sections 4.2 and 4.3. We refer the reader to the introduction of Section 3.3, where we provide a more detailed explanation regarding separation algorithms.

Since in the computational experiments which we will report in Section 4.5 we are only interested in obtaining linear programming relaxation values, we will only present exact separation algorithms without any concerns regarding efficiency and, in addition, we will not present dedicated separation algorithms for integer points. Apart from the algorithms presented in Section 4.4.1, the separation algorithms described in the other sections are based on max-flow/min-cut computations in the auxiliary st-extended graph, which we recall is the graph $W_1 = (V_1, A_1)$, originally defined in Section 3.3, and which could be obtained from the original graph G as follows:

- The set of nodes V_1 includes all nodes of V and two additional nodes s and t, that is, $V_1 = V \cup \{s, t\}$;

- The set of arcs A_1 includes all arcs of A except the arcs ingoing the depots, additional arcs linking node s to every depot $d \in D$ and additional arcs linking every client $i \in C$ to node t, that is, $A_1 = (A \setminus \{(i, d) : i \in C,\ d \in D\}) \cup \{(s, d) : d \in D\} \cup \{(i, t) : i \in C\}$.

4.4.1 Separation of constraints (4.10), (4.12) and (4.15)

The separation algorithms for constraints $v^i_{D'} + x_{ij} \leq v^j_{D'} + 1$ (4.10), $v^i_{D'} + x_{ij} + x_{ji} \leq v^j_{D'} + 1$ (4.12) and $v^i_{D'} + x_{ij} + x(D', j) + x(i, D \setminus D') \leq v^j_{D'} + 1$ (4.15) are based on a separation algorithm described by Erdoğan et al. (2016) for the Hamiltonian p-median problem.

Algorithm 4.1

Require: A point (x^*, v^*).

1: **for all** $(p, q) \in A^C$ **do**
2: Set $w^{lhs} = 0$ and $D' = \emptyset$.
3: **for all** $d \in D$ **do**
4: If $v_d^{*p} - v_d^{*q} > 0$, then set $w^{lhs} = w^{lhs} + v_d^{*p} - v_d^{*q}$ and $D' = D' \cup \{d\}$.
5: **end for**
6: **if** $w^{lhs} > 1 - x_{pq}^*$ **then**
7: D' defines a violated inequality (4.10).
8: **end if**
9: **end for**

Algorithm 4.2

Require: A point (x^*, v^*).

1: **for all** $(p, q) \in A^C$ **do**
2: Set $w^{lhs} = 0$ and $D' = \emptyset$.
3: **for all** $d \in D$ **do**
4: If $v_d^{*p} - v_d^{*q} > 0$, then set $w^{lhs} = w^{lhs} + v_d^{*p} - v_d^{*q}$ and $D' = D' \cup \{d\}$.
5: **end for**
6: **if** $w^{lhs} > 1 - x_{pq}^* - x_{qp}^*$ **then**
7: D' defines a violated inequality (4.12).
8: **end if**
9: **end for**

We start by observing that constraints (4.10), (4.12) and (4.15) can be rewritten, respectively, in the following form:

$$v_{D'}^i - v_{D'}^j \leq 1 - x_{ij} \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j \qquad (4.47)$$

$$v_{D'}^i - v_{D'}^j \leq 1 - x_{ij} - x_{ji} \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j \qquad (4.48)$$

$$v_{D'}^i - v_{D'}^j + x(D', j) + x(i, D \setminus D') \leq 1 - x_{ij} \qquad \forall D' \subset D,\ \forall i, j \in C,\ i \neq j. \qquad (4.49)$$

In order to separate these constraints we use algorithms 4.1, 4.2 and 4.3, respectively, which are polynomial-time exact separation algorithms for both fractional and integer points. Intuitively, notice that, for each arc $(i, j) \in A^C$, the right-hand side of any of the above constraints is fixed and we can maximize the left-hand side by checking for each depot $d \in D$ whether we should have $d \in D'$ or $d \in D \setminus D'$.

4.4.2 Separation of constraints (4.11)

In order to separate constraints $v_{D'}^i + x(S) \leq v_{D'}^j + |S| - 1$ (4.11), notice that if we appropriately use the client indegree constraints (1.5) for the nodes in S and constraints $\sum_{d \in D} v_d^i = 1$ (4.3)

Algorithm 4.3

Require: A point (x^*, v^*).

1: **for all** $(p, q) \in A^C$ **do**
2: Set $w^{lhs} = 0$ and $D' = \emptyset$.
3: **for all** $d \in D$ **do**
4: If $v_d^{*p} - v_d^{*q} + x_{dq}^* > x_{pd}^*$, then set $w^{lhs} = w^{lhs} + v_d^{*p} - v_d^{*q} + x_{dq}^*$ and $D' = D' \cup \{d\}$. Otherwise, set $w^{lhs} = w^{lhs} + x_{pd}^*$.
5: **end for**
6: **if** $w^{lhs} > 1 - x_{pq}^*$ **then**
7: D' defines a violated inequality (4.15).
8: **end if**
9: **end for**

Algorithm 4.4

Require: A point (x^*, v^*) and the auxiliary st-extended graph.

1: **for all** $i, j \in C,\ i \neq j$ **do**
2: Set the capacities of the arcs $\{(s, d) : d \in D\}$ to 1, the capacities of the arcs $(p, q) \in A \setminus \{(k, d) : k \in C,\ d \in D\}$ to x_{pq}^*, the capacities of the arcs (i, t) and (j, t) to 1 and the capacities of the arcs $\{(k, t) : k \in C,\ k \neq i,\ k \neq j\}$ to 0. Then, determine the maximum flow w^{lhs} from s to t and define S as the client nodes in the same shore as node t in the corresponding minimum cut.
3: Set $w^{rhs} = 0$ and $D' = \emptyset$.
4: **for all** $d \in D$ **do**
5: If $v_d^{*i} > v_d^{*j}$, then set $w^{rhs} = w^{rhs} + v_d^{*i}$ and $D' = D' \cup \{d\}$. Otherwise, set $w^{rhs} = w^{rhs} + v_d^{*j}$.
6: **end for**
7: **if** $w^{lhs} < w^{rhs}$ **then**
8: S and D' define a violated inequality (4.11).
9: **end if**
10: **end for**

for node j, we can rewrite constraints (4.11) in the following cut form:

$$x(D \cup S', S) \geq v_{D'}^i + v_{D \setminus D'}^j \qquad \forall D' \subset D,\ \forall S \subset C :\ 2 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j. \tag{4.50}$$

This way of writing the constraints shows that we can separate them by minimizing the left-hand side and maximizing the right-hand side independently, since the set S only appears on the left-hand side and the choice of set D' is only important for the right-hand side. Based on this observation, we define algorithm 4.4 which is a polynomial-time separation algorithm for both fractional and integer points. Intuitively, we minimize the cut on the left-hand side by resorting to max-flow/min-cut computations in the st-extended graph and we maximize the right-hand side by inspection, that is, by choosing for each depot $d \in D$ whether we should have $d \in D'$

Algorithm 4.5

Require: A point (x^*, v^*, z^*) and the auxiliary st-extended graph.

1: **for all** $d \in D$ and $i \in C$ **do**
2: Set the capacity of the arc (s, d) to 1, the capacities of the arcs $\{(s, d') : d' \neq d\}$ to 0, the capacities of the arcs $(d, q) \in A_O^d$ to z_{dq}^{*d}, the capacities of the arcs $(d', q) \in A_O^{d'}$, for $d' \neq d$, to 0, the capacities of the arcs $(p, q) \in A^C$ to z_{pq}^{*d}, the capacity of the arc (i, t) to 1, the capacities of the arcs $\{(k, t) : k \in C,\ k \neq i\}$ to 0, and determine the maximum flow w from s to t.
3: **if** $w < v_d^{*i}$ **then**
4: The corresponding minimum cut defines a violated inequality (4.43) for d and i in which S is the subset of client nodes in the same shore as node t.
5: **end if**
6: **end for**

or $d \in D \setminus D'$.

4.4.3 Separation of constraints (4.43)

In order to separate constraints $z^d(\{d\} \cup S', S) \geq v_d^i$ (4.43) for both fractional and integer points we can use algorithm 4.5, which is a polynomial-time exact separation algorithm based on the auxiliary st-extended graph. Intuitively, observe that the left-hand side, for each $d \in D$, is a cut which only involves z variables with respect to d, thus, we can minimize it by performing max-flow/min-cut computations in the st-extended graph in which the capacities of the arcs are given by the z variables associated with depot d.

4.4.4 Separation of constraints (4.19)

In order to separate constraints $x(S) + x(D', i) + x(j, D \setminus D') + x_{ji} \leq |S|$ (4.19), we start by noticing that, by appropriately using the client indegree constraints (1.5) for the set S, they can be rewritten as follows:

$$x(D \cup S', S) \geq x(D', i) + x(j, D \setminus D') + x_{ji}$$
$$\forall D' \subset D,\ \forall S \subset C :\ 2 \leq |S| \leq |C| - |D|,\ \forall i, j \in S,\ i \neq j. \quad (4.51)$$

Once again this shows that we can separate these constraints by minimizing the left-hand side and maximizing the right-hand side independently, since the set S only appears on the left-hand side and the choice D' is only relevant for the right-hand side. The separation algorithm we propose is algorithm 4.6. Intuitively, we minimize the cut on the left-hand side by resorting to max-flow/min-cut computations in the st-extended graph and we maximize the right-hand by choosing for each depot $d \in D$ whether we should have $d \in D'$ or $d \in D \setminus D'$.

Algorithm 4.6

Require: A point x^* and the auxiliary st-extended graph.

for all $i, j \in C,\ i \neq j$ **do**
  Set the capacities of the arcs $\{(s,d) : d \in D\}$ to 1, the capacities of the arcs $(p,q) \in A \setminus \{(k,d) : k \in C,\ d \in D\}$ to x^*_{pq}, the capacities of the arcs (i,t) and (j,t) to 1, the capacities of the arcs $\{(k,t) : k \in C,\ k \neq i,\ k \neq j\}$ to 0. Then, determine the maximum flow w^{lhs} from s to t and define S as the client nodes in the same shore as node t in the corresponding minimum cut.
  Set $w^{rhs} = x^*_{ji}$ and $D' = \emptyset$.
  for all $d \in D$ **do**
    If $x^*_{di} > x^*_{jd}$, then set $w^{rhs} = w^{rhs} + x^*_{di}$ and $D' = D' \cup \{d\}$. Otherwise, set $w^{rhs} = w^{rhs} + x^*_{jd}$.
  end for
  if $w^{lhs} < w^{rhs}$ **then**
    S and D' define a violated inequality (4.19).
  end if
end for

4.5 Computational experiment

In this section we present a set of computational results that complement the theoretical discussion of Sections 4.2 and 4.3. More precisely, we compare the linear programming relaxation values provided by some of the systems of inequalities presented throughout these sections.

We will divide this study into two parts. In Section 4.5.1 we compare systems of inequalities that model path elimination constraints defined in the space of the x and the v variables presented in Section 4.2. Then, in Section 4.5.2, we compare a subset of these previous systems to the systems based on the arc-depot assignment variables z, originally presented in Section 2.3, and, in addition, we present linear programming relaxation values corresponding to a formulation based on the ZV-CUTS system presented in Section 4.3.3.

For the ensuing experiments we will use the set T of instances that have been described in Section 3.4. The reason for using a set of instances with a small size is due to the fact that many of the proposed modeling approaches being compared have too many variables and/or constraints and, thus, cannot be used to solve large instances in a reasonable computational time. The linear programming relaxation values reported in the following sections are the real linear programming relaxation values, that is, we deactivate all of CPLEX's general purpose cuts and preprocessing. Note also that the optimal value of these instances is known since it was determined by using the branch-and-cut algorithm presented in Chapter 3.

4.5.1 Comparing path elimination constraints in the space of the x and the v variables

In this section we compare, in terms of the corresponding linear programming relaxation values, several of the proposed systems of inequalities in the space of the x and the v variables to model path elimination constraints which were presented throughout Section 4.2. More precisely, we define valid formulations in the space of the x and the v variables for the multi-depot routing problem that only differ in the set of path elimination constraints used, which allows us to compare the several systems of path elimination constraints in a fair way.

Consider the base formulation comprised of the degree constraints (1.2)–(1.5), the subtour elimination constraints $x(D \cup S', S) \geq 1$ (2.2), the domain constraints for the x variables (1.8), and the following constraints in the space of the x and the v variables, which are common to all of the different systems of inequalities of Section 4.2:

$$x_{di} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.1}$$

$$x_{id} \leq v_d^i \qquad \forall d \in D,\ \forall i \in C \tag{4.2}$$

$$\sum_{d \in D} v_d^i = 1 \qquad \forall i \in C \tag{4.3}$$

$$v_d^i \in \{0, 1\} \qquad \forall d \in D,\ \forall i \in C. \tag{4.5}$$

We will compare a total of five formulations. Each formulation is comprised of the base formulation presented above with the addition of a set of constraints. For simplification, we denote the formulations by the same name of the system of inequalities in the space of the x and the v variables from which it originates. The formulations which will be tested are the following:

- The CDA-DDL formulation which uses constraints $v_d^i + x_{ij} + x_{ji} \leq v_d^j + 1$ (4.8);
- The CDA$^+$ formulation which uses constraints $v_d^i + x_{ij} + x_{dj} + x(i, D \setminus \{d\}) \leq v_d^j + 1$ (4.16);
- The GCDA-DDL formulation which uses constraints $v_{D'}^i + x_{ij} + x_{ji} \leq v_{D'}^j + 1$ (4.12);
- The GCDA$^+$ formulation which uses constraints $v_{D'}^i + x_{ij} + x(D', j) + x(i, D \setminus D') \leq v_{D'}^j + 1$ (4.15);
- The GCDA-GDDL formulation which uses constraints $v_{D'}^i + x(S) \leq v_{D'}^j + |S| - 1$ (4.11).

Tables 4.1 and 4.2 show the linear programming relaxation values of the five formulations which are being compared, with Table 4.1 focusing on the asymmetric instances of the instance

Table 4.1: Comparing the linear programming relaxation values of several path elimination constraints based on the v variables for asymmetric instances

Name	OPT	CDA-DDL	CDA+	GCDA-DDL	GCDA+	GCDA-GDDL
t-bgs-50-20a	821	812.714	818.584	815.031	820	815.631
t-bgs-50-19a	824	815.714	821.573	818.017	823	818.631
t-bgs-50-18a	807	799.75	804.067	800.791	804.519	800.853
t-bgs-50-17a	784	779.396	782.636	780.25	784	780.25
t-bgs-50-16a	752	747.396	750.636	748.25	752	748.25
t-bgs-50-15a	754	744.146	744.636	744.963	746	745.217
t-bgs-50-14a	757	742.093	742.051	743.057	743.917	743.994
t-bgs-50-13a	749	730.266	730.181	731.568	732.782	732.664
t-bgs-50-12a	741	722.257	722.131	723.517	724.656	724.311
t-bgs-50-11a	722	710.25	710.273	711.332	713.158	711.531
t-bgs-50-10a	719	705.364	705.5	705.9	707.25	705.9
t-bgs-50-9a	707	692.962	694.5	692.962	694.5	692.962
t-bgs-50-8a	689	676.815	678.554	676.815	678.563	676.951
t-bgs-50-7a	679	670	670	670	670	670
t-bgs-50-6a	659	651.167	651.167	651.167	651.167	651.167
t-bgs-50-5a	648	639.75	639.75	639.75	639.75	639.75
t-bgs-50-4a	621	610.563	610.563	610.563	610.563	610.563
t-bgs-50-3a	628	620.846	620.846	620.846	620.846	620.846
t-bgs-50-2a	628	619.75	619.75	619.75	619.75	619.75

set T and Table 4.2 on the symmetric ones. Both tables have the following format. The first column indicates the instance name and the second column its optimal value (OPT). The remaining five columns indicate the linear programming relaxation value of formulations CDA-DDL, CDA$^+$, GCDA-DDL, GCDA$^+$ and GCDA-GDDL, respectively.

First we analyze the results in Table 4.1, which are with respect to the asymmetric instances. Observe that all five formulations provided the same linear programming relaxation value for the instances with between 2 and 7 depots. Regarding the instances with 8 or more depots, the results show that, between the two compact systems of path elimination constraints, namely the CDA-DDL system and the CDA$^+$ system, the latter provides a higher linear programming relaxation value than the former except in the instances with 12, 13 and 14 depots. The conclusions are similar when we compare their depot subset generalizations, namely systems GCDA-DDL and GCDA$^+$, respectively, but now the latter provides a higher linear programming relaxation value than the former in all instances with at least 8 depots. In any of the two cases, the depot subset generalized systems are able to improve the linear programming relaxation values of the corresponding compact systems. Finally, the generalization of the GCDA-DDL system to arc

Table 4.2: Comparing the linear programming relaxation values of several path elimination constraints based on the v variables for symmetric instances

Name	OPT	CDA-DDL	CDA+	GCDA-DDL	GCDA+	GCDA-GDDL
t-bgs-50-20	1362	1324	1324	1324	1324	1324
t-bgs-50-19	1367	1325	1325	1325	1325	1325
t-bgs-50-18	1342	1300	1300	1300	1300	1300
t-bgs-50-17	1311	1283.5	1283.5	1283.5	1283.5	1283.5
t-bgs-50-16	1263	1236.5	1236.5	1236.5	1236.5	1236.5
t-bgs-50-15	1251	1234.5	1234.5	1234.5	1234.5	1234.5
t-bgs-50-14	1241	1224.5	1224.5	1224.5	1224.5	1224.5
t-bgs-50-13	1229	1217.5	1217.5	1217.5	1217.5	1217.5
t-bgs-50-12	1232	1220.5	1220.5	1220.5	1220.5	1220.5
t-bgs-50-11	1229	1198	1198	1198	1198	1198
t-bgs-50-10	1188	1163	1163	1163	1163	1163
t-bgs-50-9	1193	1166	1166	1166	1166	1166
t-bgs-50-8	1170	1145.5	1145.5	1145.5	1145.5	1145.5
t-bgs-50-7	1159	1137.5	1137.5	1137.5	1137.5	1137.5
t-bgs-50-6	1128	1113	1113	1113	1113	1113
t-bgs-50-5	1105	1091.65	1091.65	1091.65	1091.65	1091.65
t-bgs-50-4	1053	1050.25	1050.25	1050.25	1050.25	1050.25
t-bgs-50-3	1077	1073.88	1073.88	1073.88	1073.88	1073.88
t-bgs-50-2	1078	1074.75	1074.75	1074.75	1074.75	1074.75

sets defining a clique, namely the system GCDA-GDDL, slightly improves the linear programming relaxation values over the GCDA-DDL system and it is now marginally better than the GCDA$^+$ system in the instance with 14 depots.

Overall the results for asymmetric instances indicate that the CDA$^+$ and the GCDA$^+$ systems, which were derived from the 3I^{++} system in Section 4.2.6, seem to be more effective than the CDA-DDL, the GCDA-DDL and the GCDA-GDDL systems, which are based on arguments similar to ones used in the precedence constrained traveling salesman problem. Recall, however, that the GCDA-GDDL system, which includes constraints (4.11), implies the subtour elimination constraints $x(S) \leq |S| - 1$ (2.1) as we showed in Section 4.2.3, which is an advantage since we do not need separate them as well. Additionally, the results of Table 4.1 also clearly show that the compact systems provide, in general, a smaller linear programming relaxation value than their depot subset and/or arc set generalized versions, however, the latter require separation algorithms to use and, in some cases, the improvement is not substantial or even non-existent.

Concerning the results of Table 4.2, which are with respect to the symmetric instances, we

can see that all formulations provide the same linear programming relaxation value for all instances. Regarding the formulations CDA$^+$ and GCDA$^+$, this is related to the result of Proposition 3, from which we could conclude that the linear programming relaxation values for symmetric instances of any system of path elimination constraints which is implied by the 3I^{++} system is always the same. Note that we are unaware if a similar result to the one of Proposition 3 applies to the CDA-DDL, the GCDA-DDL or the GCDA-GDDL systems, even if the results of Table 4.2 seem to indicate so. Overall, the conclusions we can draw from these results are limited and, in fact, it is not clear which system of inequalities is preferable to use for symmetric cost instances.

4.5.2 Comparing formulations using depot assignment variables

In this section we assess the linear programming relaxation values provided by formulations based on the arc-depot assignment variables z, and compare them to each other and to a subset of the formulations defined in the previous section, which used path elimination constraints in the space of the x and the v variables. The formulations which will be compared in this section consider as a base formulation the degree constraints (1.2)–(1.5), the subtour elimination constraints $x(D \cup S', S) \geq 1$ (2.2) and the domain constraints for the x variables (1.8). Once again, for simplification, we denote the formulations by the same name of the system of inequalities from which it originates.

We consider two formulations from the previous section, namely the GCDA$^+$ formulation and the GCDA-GDDL formulation. These two formulations will be compared to the 3I formulation, which uses the 3I system presented in Section 2.3.1 and which we recall is as follows

$$\sum_{j \in C} z^d_{dj} = 1 \qquad \forall d \in D \tag{2.3}$$

$$\sum_{j \in C} z^d_{jd} = 1 \qquad \forall d \in D \tag{2.4}$$

$$\sum_{j \in \{d\} \cup C} z^d_{ji} = \sum_{j \in \{d\} \cup C} z^d_{ij} \qquad \forall d \in D,\ \forall i \in C \tag{2.5}$$

$$z^d_{ij} \leq x_{ij} \qquad \forall d \in D,\ \forall (i,j) \in A^C \cup A^d_O \cup A^d_I \tag{2.6}$$

$$z^d_{ij} \in \{0,1\} \qquad \forall d \in D,\ \forall (i,j) \in A^C \cup A^d_O \cup A^d_I, \tag{2.7}$$

the 3I^{++} formulation, which uses the 3I^{++} system that can be obtained from the one above by replacing the linking constraints between the z and the x variables (2.6) by the set

$$\sum_{d \in D} z^d_{ij} = x_{ij} \qquad \forall (i,j) \in A^C \tag{2.10}$$

$$z^d_{ij} = x_{ij} \qquad \forall d \in D,\ \forall (i,j) \in A^d_O \cup A^d_I, \tag{2.11}$$

Table 4.3: Comparing the linear programming relaxation values of formulations based on the v variables and formulations based on the z variables for asymmetric instances

Name	OPT	GCDA+	GCDA-GDDL	3I	3I^{++}	ZV-CUTS
t-bgs-50-20a	821	820	815.631	819.9	820	820
t-bgs-50-19a	824	823	818.631	822.9	823	823
t-bgs-50-18a	807	804.519	800.853	805.4	805.4	806.1
t-bgs-50-17a	784	784	780.25	784	784	784
t-bgs-50-16a	752	752	748.25	752	752	752
t-bgs-50-15a	754	746	745.217	746	746	748.5
t-bgs-50-14a	757	743.917	743.994	748	748	750.5
t-bgs-50-13a	749	732.782	732.664	737.286	738.875	740.375
t-bgs-50-12a	741	724.656	724.311	729.188	730.5	732.333
t-bgs-50-11a	722	713.158	711.531	715.714	717.565	718.125
t-bgs-50-10a	719	707.25	705.9	708.038	709.638	710.664
t-bgs-50-9a	707	694.5	692.962	695.531	696.467	697.903
t-bgs-50-8a	689	678.563	676.951	679	679.613	682.247
t-bgs-50-7a	679	670	670	670	670	671.25
t-bgs-50-6a	659	651.167	651.167	651.167	651.167	652
t-bgs-50-5a	648	639.75	639.75	639.75	639.75	640.4
t-bgs-50-4a	621	610.563	610.563	610.563	610.563	611.333
t-bgs-50-3a	628	620.846	620.846	620.846	620.846	621
t-bgs-50-2a	628	619.75	619.75	619.75	619.75	620.5

and the ZV-CUTS formulation, which uses the ZV-CUTS system defined by the 3I^{++} system, constraints $z^d(\{d\} \cup S', S) \geq v^i_d$ (4.43), the relationships between the v and the z variables (4.13)–(4.14), and the domain constraints for the v variables (4.5).

Tables 4.3 and 4.4 show the linear programming relaxation values of the five formulations which are being compared, with Table 4.3 focusing on the asymmetric instances of the instance set T and Table 4.4 on the symmetric ones. Both tables have the following format. The first column indicates the instance name and the second column its optimal value (OPT). The remaining five columns indicate the linear programming relaxation value of formulations GCDA$^+$, GCDA-GDDL, 3I, 3I^{++} and ZV-CUTS, respectively. Observe that the results for the formulations GCDA$^+$ and GCDA-GDDL were taken from Tables 4.1 and 4.2 and the results for the formulation 3I were taken from Table 3.1, since we have shown that the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ are equivalent to the 3I system in Proposition 5.

We first analyze the results of Tables 4.3 and 4.4 by ignoring the last column, which concerns the ZV-CUTS formulation. The results in Table 4.3, which are with respect to the asymmetric instances of the instance set T, show that, in general, the arc-depot assignment variable based

Table 4.4: Comparing the linear programming relaxation values of formulations based on the v variables and formulations based on the z variables for symmetric instances

Name	OPT	GCDA+	GCDA-GDDL	3I	3I^{++}	ZV-CUTS
t-bgs-50-20	1362	1324	1324	1324	1324	1357.5
t-bgs-50-19	1367	1325	1325	1325	1325	1362.5
t-bgs-50-18	1342	1300	1300	1300	1300	1335.5
t-bgs-50-17	1311	1283.5	1283.5	1283.5	1283.5	1308.5
t-bgs-50-16	1263	1236.5	1236.5	1236.5	1236.5	1263
t-bgs-50-15	1251	1234.5	1234.5	1234.5	1234.5	1251
t-bgs-50-14	1241	1224.5	1224.5	1224.5	1224.5	1241
t-bgs-50-13	1229	1217.5	1217.5	1217.5	1217.5	1229
t-bgs-50-12	1232	1220.5	1220.5	1220.5	1220.5	1232
t-bgs-50-11	1229	1198	1198	1198	1198	1224
t-bgs-50-10	1188	1163	1163	1163	1163	1186
t-bgs-50-9	1193	1166	1166	1166	1166	1191
t-bgs-50-8	1170	1145.5	1145.5	1145.5	1145.5	1170
t-bgs-50-7	1159	1137.5	1137.5	1137.5	1137.5	1159
t-bgs-50-6	1128	1113	1113	1113	1113	1128
t-bgs-50-5	1105	1091.65	1091.65	1091.65	1091.65	1105
t-bgs-50-4	1053	1050.25	1050.25	1050.25	1050.25	1053
t-bgs-50-3	1077	1073.88	1073.88	1073.88	1073.88	1076.25
t-bgs-50-2	1078	1074.75	1074.75	1074.75	1074.75	1077.75

systems provide higher linear programming relaxation values than the best of the two systems defined in the space of the x and the v variables. Regarding the 3I^{++} system, this is only not true for the instances with between 2 and 7 depots in which this formulation provides the same linear programming relaxation value as the ones defined in the space of the x and the v variables. As for the 3I formulation, its linear programming relaxation value is marginally worse than that of the GCDA^{+} formulation for the instances with 19 and 20 depots but, for the remaining cases, it is either the same or higher. Additionally, the results also show that the 3I^{++} system is only marginally better than the 3I system.

Given that the 3I system is equivalent, in terms of the corresponding linear programming relaxation, to the 1-MCC inequalities (2.13) and the 3I^{++} system implies the more general k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14), the latter proved in Proposition 6, we can conclude that the k-MCC inequalities (2.14) are only marginally better than the 1-MCC inequalities (2.13) in terms of linear programming relaxation. Note also that, due to the result of Proposition 5, we can see that the 1-MCC inequalities (which can be separated in a very efficient way) provide, in general, as good as or even higher linear programming relaxation values when

compared to the path elimination constraint systems defined in the space of the x and the v variables and, thus, it is unlikely that the results of the branch-and-cut algorithm presented in Chapter 3 could be strengthened by using the systems in the space of the x and the v variables which we have proposed in this chapter.

As for the results in Table 4.4, once again we see that the linear programming relaxation values of the two systems in the space of the x and the v variables and of the 3I and the 3I^{++} system are the same, which was expected given the result of Proposition 3.

We now analyze in detail the results of Tables 4.3 and 4.4 regarding the last column, that is, regarding the ZV-CUTS system. Note that the difference in the linear programming relaxation values when compared to the 3I^{++} system is only due to constraints (4.43), since the linking constraints between the v and the z variables (4.13)–(4.14) provide no increase in the linear programming relaxation value on their own. As we mentioned when we first introduced this system in Section 4.3.3, the linear programming relaxation values obtained are very close to the optimal value of the instances tested. For the asymmetric instances in Table 4.3, we can see that the ZV-CUTS system is able to improve the linear programming relaxation values in most of the instances and, in particular, it is the only system which could obtain an improvement on the instances with between 2 and 7 depots. However, the improvements are not substantial, which could be due to the fact that the 3I^{++} system is already good for asymmetric instances. In contrast, in the symmetric instances of Table 4.4, the ZV-CUTS system provides outstanding results. Firstly, it is the first system for which the linear programming relaxation values do not coincide with all the other systems. Secondly, the improvement on the linear programming relaxation values is very significant with the ZV-CUTS system being able to obtain the optimal solution of 10 out of 19 instances, and on the remaining ones providing linear programming relaxation values which are very close to the optimal value. Overall, the ZV-CUTS system provides high linear programming relaxation values for any instance and it is definitely a formulation which should be studied in more detail in the future.

4.6 Concluding remarks

In this chapter we presented several systems of inequalities to model path elimination constraints as complement to the study of the previous chapters.

Most of the systems of inequalities presented were based on client-depot assignment variables similar to the precedence variables used in the precedence constrained traveling salesman problem. The advantage of these variables is that they allow for intuitive ways of modeling constraints, in particular path elimination constraints. Some of these systems were based on ideas from the precedence constrained traveling salesman problem and from the Hamiltonian p-median problem, which is the problem studied in the second part of this dissertation. The

remaining systems were based on a relationship to the arc-depot assignment variables first used in Chapter 2 and are, as far as we know, new. The computational results conducted in this chapter showed that the latter type of systems usually provide higher linear programming relaxation values than the former for asymmetric instances, even though they are theoretically not comparable.

We also presented another system of inequalities to model path elimination constraints based on double network flow systems. The two flow systems work together to ensure that flow is sent from each depot to each client assigned to that depot and back from the client to the depot and were first linked with the arc variables. By linking these flow systems with the arc-depot assignment variables, we were able to derive a stronger system of inequalities which also models subtour elimination constraints. Additionally, based on two different results we were able to show that we can derive a non-compact system of inequalities which does not require the complex double network flows system but which provides the same linear programming relaxation value. The computational results showed that a formulation based on this new system is able to provide linear programming relaxation values which are very close to the optimal value in the instances tested.

Part II

The Hamiltonian p-median problem

Chapter 5

Introducing the Hamiltonian p-median problem

Contents

5.1 Introduction

In this second part of the dissertation we study the Hamiltonian p-median problem (see, e.g., Branco & Coelho 1990, Gollowitzer et al. 2014, Erdoğan et al. 2016, Bektaş, Gouveia & Santos 2019), which is the problem of finding p cycles (circuits) in an undirected (directed) graph such that each node of the graph is in one and only one cycle (circuit) and the total cost of the selected edges (arcs) is minimized. Notice that if $p = 1$ then the Hamiltonian p-median problem is the classical symmetric (asymmetric) traveling salesman problem (see, e.g., Lawler et al. 1985, Applegate et al. 2006). We will use formulations based on directed graphs, similarly to what we did for the multi-depot routing problem in the first part of this dissertation, since we assume that the cost function may be asymmetric, hence, from now on our aim is to find a set of p circuits of minimum total cost such that each node is in one and only one circuit.

As we mentioned, the traveling salesman problem is a particular case of the Hamiltonian p-median problem, however, the literature related to the Hamiltonian p-median problem is not as extensive as the one related to the traveling salesman problem. To the best of our knowledge, the Hamiltonian p-median problem was first introduced by Branco & Coelho (1990) who presented two formulations, one a set partitioning formulation and the other a formulation based on directed graphs, similar to the one by Fisher & Jaikumar (1981) for the vehicle routing problem, as well as several heuristics, assuming symmetric costs. Other studies include that of Glaab & Pott (2000) and Zohrehbandian (2007), either of which do not include computational experiments, and more recently by Hupp & Liers (2013) and Gollowitzer et al. (2014) concerning polyhedral studies. Most of the algorithmic work on this problem is fairly recent (see, e.g., Gollowitzer et al. 2014, Erdoğan et al. 2016, Marzouk, Moreno-Centeno & Üster 2016) and introduces an additional condition which is that the Hamiltonian p-median problem does not admit solutions that include two-node circuits. Clearly, this condition may lead to sub-optimal solutions when compared to the Hamiltonian p-median problem as first defined by Branco & Coelho (1990), however, it is attractive for modeling purposes, since the problem can then be easily modeled by using formulations based on undirected graphs with only a binary variable associated with each edge. Modeling two-node cycles on an undirected graph, although not straightforward, is still possible by using {0,1,2} variables or, equivalently, an additional set of binary variables that specifically considers this case, as it is done in routing problems with multiple depots or in single-depot problems with multiple vehicles (see, e.g., Laporte, Nobert & Pelletier 1983, Laporte et al. 1986, Araque G., Kudva, Morin & Pekny 1994, Belenguer et al. 2011, Benavent & Martínez-Sykora 2013).

The Hamiltonian p-median problem is related to the multi-depot routing problem that we studied in the first part of this dissertation. Observe that the nodes of the underlying graph are all identical in the Hamiltonian p-median problem, unlike what happens in the multi-depot routing problem in which nodes are partitioned into depots and clients, however, we can artificially attribute different roles to different nodes by using adequate modeling techniques. In particular, we will introduce the concept of acting depot of a circuit, that is, in each of the p circuits one of its nodes will function as an artificial depot whereas the other nodes will be considered artificial clients. By using this concept we can effectively use techniques from routing problems with multiple depots in the Hamiltonian p-median problem, even if these depots are artificial. The concept of acting depot is not new in the Hamiltonian p-median problem (see, e.g., Gollowitzer et al. 2014, Erdoğan et al. 2016) neither in other related settings (see, e.g., Laporte et al. 1983), however, we believe it can be further explored and, in particular, we believe that a stronger link can be established between routing problems with multiple depots and the Hamiltonian p-median problem.

Our approach for studying the Hamiltonian p-median problem in this dissertation follows

the paradigm given by Gollowitzer et al. (2014) in that we present models for the Hamiltonian p-median problem by partitioning the constraint set of the problem into two, namely one that guarantees that there are at most p circuits in the solution, and another to ensure that there are at least p circuits in the solution, which we will refer to as $(\leq p)$ constraints and $(\geq p)$ constraints, respectively. At this point, it is interesting to revisit the special case of the traveling salesman problem, where one wishes to obtain solutions with $p = 1$ circuit, for which reason one would need to eliminate solutions with more than $p = 1$ circuits. This leads to the observation that the $(\leq p)$ constraints are generalizations of (and similar to) the subtour elimination constraints, and that the $(\geq p)$ constraints are not as straightforward to characterize since they are unnecessary in the traveling salesman problem. By introducing the concept of acting depot, observe that the $(\leq p)$ and $(\geq p)$ constraints can be restated in the following way. On the one hand, if a solution has more than p circuits, then surely one of the circuits cannot have any acting depot. On the other hand, if a solution has less than p circuits, then one of the circuits must have two or more acting depots. Therefore, by introducing the acting depot concept, we can relate the $(\geq p)$ constraints to path elimination constraints which are more commonly associated with routing problems with multiple depots.

The main purpose of the second part of this dissertation is to present a new formulation and a corresponding branch-and-cut algorithm for the Hamiltonian p-median problem. More precisely, we explore the concept of acting depot and define a new formulation based on the idea that the arcs of the underlying graph are interpreted differently depending on whether one of its endpoints is an acting depot or not. With this novel idea we are able to provide sets of $(\leq p)$ and $(\geq p)$ constraints which can be separated in polynomial time and deal with symmetry issues arising in formulations which use the concept of acting depot. In particular, the $(\geq p)$ constraints of the new formulation are a non-straightforward adaptation of the new multi-cut constraints proposed in Section 2.4 of the first part of this dissertation and, thus, another of the contributions of this part of the dissertation is to show that adaptations of the multi-cut constraints can be used as path elimination constraints in other problems in an effective way. Additionally, we compare the new proposed formulation to an adaptation of a formulation presented by Erdoğan et al. (2016), which uses (acting-)client-(acting-)depot assignment variables similar to the ones used in Chapter 4 in the context of the multi-depot routing problem.

The rest of this chapter is organized in the following way. In Section 5.2 we formally define the Hamiltonian p-median problem and present some notation. In Section 5.3 we present a valid generic integer linear programming formulation for the problem based on a set of arc variables. Finally, in Section 5.4 we present another valid integer linear programming formulation for the Hamiltonian p-median problem that uses the same arc variables and an additional set of variables which differentiate the so-called acting depot nodes from the acting client nodes.

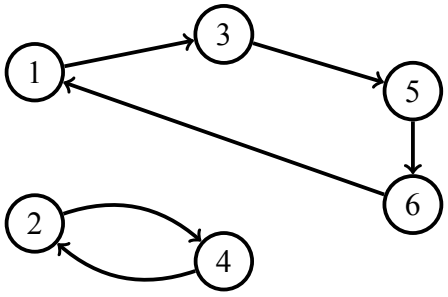

Figure 5.1: An example of a feasible solution of a Hamiltonian 2-median problem

5.2 Definitions and notation

We define the Hamiltonian p-median problem on a directed graph $G = (V, A)$, with a set of nodes $V = \{1, \ldots, n\}$, a set of arcs $A = \{(i, j) : i, j \in V,\ i \neq j\}$ and a non-negative cost function c associated with the set of arcs. Whenever necessary we will re-use some of the names used in the first part of this dissertation to define mathematical elements (such as G, V, A and c) for two reasons. Firstly it makes it easier to immediately associate the nomenclature with what it represents and, secondly, it is clear that they now refer to the Hamiltonian p-median problem. In addition, we assume that G is a complete graph, however, once again, this study is applicable to incomplete graphs by simply not considering the pairs (i, j) such that $(i, j) \notin A$ in any of the mathematical expressions.

The objective of the Hamiltonian p-median problem is to find a minimum cost set of p circuits such that each node of V is in exactly one of the circuits. Figure 5.1 shows an example of a feasible solution of a Hamiltonian p-median problem in which $V = \{1, 2, 3, 4, 5, 6\}$ and $p = 2$. In the solution, nodes 1, 3, 5 and 6 are in one circuit and nodes 2 and 4 are in another circuit.

We will use the same notation as in the first part of this dissertation to simplify the mathematical expressions, that is:

- For any general one-index variable u, we write $u(S) = \sum_{i \in S} u_i$;
- For any general two-index variable v in which both indexes are subscripts, we write $v(S) = \sum_{i,j \in S} v_{ij}$ and $v(S_1, S_2) = \sum_{i \in S_1, j \in S_2} v_{ij}$;
- For any general two-index variable w with one subscript index and one superscript index, we write $w_{S_1}^{S_2} = \sum_{i \in S_1, j \in S_2} w_i^j$;
- For any general three-index variable z with two subscript indexes and one superscript index, we write $z^k(S) = \sum_{i,j \in S} z_{ij}^k$ and $z^k(S_1, S_2) = \sum_{i \in S_1, j \in S_2} z_{ij}^k$;
- In the expressions above, for any singleton set $\{i\}$ we write i instead of $\{i\}$;
- Finally, we define $S' = K \setminus S$ for any subset of nodes $S \subseteq K \subseteq V$.

5.3 A generic model in the space of the arc variables

The Hamiltonian p-median problem can be modeled as an integer linear programming problem by using a set of binary variables $x_{ij} = 1$ if arc $(i, j) \in A$ is used in one of the circuits, and $x_{ij} = 0$ otherwise. The following model is a valid generic integer linear programming model for the Hamiltonian p-median problem:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.1}$$

$$\text{subject to:} \sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{5.2}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in V \tag{5.3}$$

$$\{(i, j) \in A : x_{ij} = 1\} \text{ contains at most } p \text{ circuits} \tag{5.4}$$

$$\{(i, j) \in A : x_{ij} = 1\} \text{ contains at least } p \text{ circuits} \tag{5.5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A. \tag{5.6}$$

The system of inequalities (5.2)–(5.3) and (5.6) is the integer linear programming formulation of the assignment problem (see, e.g., Wolsey 1998). Constraints (5.2) ensure that the outdegree of any node must be 1, whereas constraints (5.3) guarantee that the indegree of any node must be 1. Clearly any solution that satisfies these two conditions is comprised of a set of disjoint circuits, however, there is no guarantee that the number of circuits is exactly p. Thus, in order to obtain a valid model for the Hamiltonian p-median problem we require two additional sets of constraints, namely constraints (5.4) which ensure that the number of circuits is less than or equal to p and constraints (5.5) which guarantee that the number of circuits is greater than or equal to p.

This formulation is an adaptation to directed graphs of a formulation described by Gollowitzer et al. (2014). In their formulation, constraints (5.4) are modeled as generalizations of subtour elimination constraints known from the traveling salesman problem, whereas constraints (5.5) are modeled as cycle-elimination constraints. It was proved by Gollowitzer et al. (2014), however, that the separation of both sets of constraints is NP-hard, and we have no reason to believe otherwise for the case of their adaptation to directed graphs, thus, we are required to resort to heuristic separation algorithms. In addition, computational tests conducted by Gollowitzer et al. (2014), Erdoğan et al. (2016) and Marzouk et al. (2016) showed that such formulations were unable to solve many instances with reasonable sizes. This motivates the study of solution methods for the Hamiltonian p-median problem based on formulations that use additional sets of variables. In particular, and as mentioned before, we will introduce the concept of acting depot in the next section and discuss a different valid generic integer linear programming model for the Hamiltonian p-median problem which uses a new set of acting depot variables alongside

the arc variables x.

5.4 A model in the space of the arc and of the acting depot variables

In this section we present and discuss a generic model for the Hamiltonian p-median problem which differs from the one in Section 5.3 in that we use an additional set of acting depot variables. More precisely, we consider the arc variables x defined previously and another set of binary acting depot variables $y_i = 1$ if node $i \in V$ is an acting depot, and $y_i = 0$ otherwise. For simplicity, we will refer to a node $i \in V$ as an acting depot or just depot if $y_i = 1$, and as an acting client or just client if $y_i = 0$, in a given solution. Consider the following model:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.1}$$

$$\text{subject to:} \sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{5.2}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in V \tag{5.3}$$

$$\sum_{i\in V} y_i = p \tag{5.7}$$

$$\{(i,j) \in A : x_{ij} = 1 \text{ and } i \in V : y_i = 1\}$$
$$\text{contains no circuit with zero depots} \tag{5.8}$$

$$\{(i,j) \in A : x_{ij} = 1 \text{ and } i \in V : y_i = 1\}$$
$$\text{contains no circuit with two or more depots} \tag{5.9}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{5.6}$$

$$y_i \in \{0,1\} \qquad \forall i \in V. \tag{5.10}$$

This formulation is a valid generic integer linear programming formulation for the Hamiltonian p-median problem. Any solution which satisfies the degree constraints (5.2)–(5.3) and constraints (5.7), which ensure that there are exactly p depots in any feasible solution, must be comprised of a set of disjoint circuits in which p nodes of V are acting depots. Notice that there is no structural difference in these solutions when compared to the solutions which only satisfy the degree constraints (5.2)–(5.3), however, by introducing these acting depots, we can rewrite the generic $(\leq p)$ constraints (5.4) and the generic $(\geq p)$ constraints (5.5) in a different, and more familiar, way. Observe that if there exists a solution with more than p circuits then at least one of the circuits has zero depots, and if there exists a solution with less than p circuits then at least one of the circuits has more than one depot. Both of these unfeasible cases are prevented, respectively, by the generic constraints (5.8) and (5.9).

In order to model both generic sets (5.8) and (5.9) we will establish a connection to the multi-depot routing problem studied in the first part of this dissertation. Recall that, in the multi-depot routing problem, in order to prevent the existence of circuits with zero depots we used subtour elimination constraints, whereas in order to prevent the existence of circuits with two or more depots we used path elimination constraints. By using the acting depot concept, we can adapt constraints used in the context of the multi-depot routing problem to the context of the Hamiltonian p-median problem. However, the adaptation is not straightforward since the set of depots is no longer fixed.

We shall illustrate this connection by presenting a valid formulation for the Hamiltonian p-median problem defined in the space of the x and the y variables. In particular, in Section 5.4.1 we present a set of $(\leq p)$ constraints, in Section 5.4.2 we present a set of $(\geq p)$ constraints and in Section 5.4.3 we present the complete proposed formulation in the space of the x and the y variables and identify some of its drawbacks which will motivate the remainder of the study in the second part of this dissertation.

5.4.1 Modeling the $(\leq p)$ constraints in the space of the x and the y variables

With respect to the generic constraints (5.8), which we recall can be seen as subtour elimination constraints in the acting depot context, we can use a set of constraints which correspond to the directed version of a set of constraints proposed by Laporte et al. (1983) (see also Gollowitzer et al. 2014), which are based on the observation that if there is no acting depot in a given set $S \in V$, that is, if $y(S) = 0$, then a circuit composed only of nodes of S cannot exist:

$$x(S) \leq |S| - 1 + y(S) \qquad \forall S \subset V : \; 2 \leq |S| \leq |V| - p. \tag{5.11}$$

Observe that if the set of depots is fixed, then the above constraints are precisely the subtour elimination constraints $x(S) \leq |S| - 1$ originally proposed by Dantzig et al. (1954) for the traveling salesman problem.

5.4.2 Modeling the $(\geq p)$ constraints in the space of the x and the y variables

Recall that the generic constraints (5.9) may be seen as path elimination constraints in the acting depot setting. Thus, in order to model the $(\geq p)$ constraints in the space of the x and the y variables, we propose an adaptation of the new multi-cut constraints presented for the multi-depot routing problem. More precisely, we show how to adapt the basic 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) first presented in Section 2.4.1.

Intuitively, consider a partition into four subsets of the set of nodes V, namely sets S', S, I and $\{i\}$, with $|I| = p - 1$. Suppose that the nodes in $I \cup \{i\}$ are the p acting depots and that the nodes in S' and S, which form a partition of $V \setminus (I \cup \{i\})$, are the acting clients. A set of path elimination constraints in this case should ensure that nodes i and j, for any acting depot $j \in I$ (thus, different from i), are in different circuits. Note, however, that this interpretation assumes that the nodes in $I \cup \{i\}$ are depots and that the nodes in $V \setminus (I \cup \{i\})$ are clients. In the multi-depot routing problem there was a clear distinction between depot nodes and clients nodes, however, in the Hamiltonian p-median problem the nodes in $I \cup \{i\}$ may not be depots and, thus, a path between i and j may be feasible. One way of imposing this condition and, consequently, defining a valid set of path elimination constraints using the x and the y variables is as follows:

$$x(S', i) + x(S', S) + x(i, S) \geq y_i + y(I) - |I| \\ \forall i \in V,\ \forall I \subset V \setminus \{i\} :\ |I| = p - 1,\ \forall S \subset V \setminus (I \cup \{i\})\,. \qquad (5.12)$$

Observe that constraints (5.12) are redundant unless all nodes of $I \cup \{i\}$ are depots, which is exactly the requirement we needed to impose. Then, the validity of these constraints follows from similar arguments as the ones used in the proof of Proposition 4, where we showed that the 1-MCC inequalities (2.13) were valid for the multi-depot routing problem.

By appropriately using the degree constraints (5.2)–(5.3), we can rewrite constraints (5.12) in the following alternative form:

$$y(I) + x(I, S) + x(S) + x(S, i) + y_i + x(I, i) \leq |S| + 1 + |I| \\ \forall i \in V,\ \forall I \subset V \setminus \{i\} :\ |I| = p - 1,\ \forall S \subset V \setminus (I \cup \{i\})\,. \qquad (5.13)$$

In this form we can see that constraints (5.13) become $y(I) + x(I, i) + y_i \leq 1 + |I|$ if $S = \emptyset$ and, thus, they ensure that there can be no arcs between acting depots, which is a particular case of an unfeasible path between depots which did not occur in the multi-depot routing problem since arcs between depots did not exist.

We can thus conclude that constraints (5.12) provide a valid representation of the generic ($\geq p$) constraints (5.9). Observe that a constraint similar to constraints (5.12) but in which $|I \cup \{i\}| > p$ would be redundant as well, since the right-hand side would never be greater than zero, however, we believe that the cases where $|I \cup \{i\}| < p$ might not be redundant in the associated linear programming relaxation. In fact, the situation where $I = \{j\}$ is related to constraints known from the literature. More precisely, constraints similar to (5.13) in this case are a stronger version of the path elimination constraints

$$y_i + x(P_{ij}) + y_j \leq |P_{ij}| \qquad \forall i, j \in V,\ i \neq j, \qquad (5.14)$$

where P_{ij} is an elementary path from i to j and $|P_{ij}|$ is the number of nodes in the path.

5.4.3 The complete model

By using the $(\leq p)$ constraints of Section 5.4.1 and the $(\geq p)$ constraints of Section 5.4.2, we obtain a valid formulation for the Hamiltonian p-median problem in the space of the x and the y variables. For clarity, we present it here:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.1}$$

$$\text{subject to:} \sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{5.2}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in V \tag{5.3}$$

$$\sum_{i\in V} y_i = p \tag{5.7}$$

$$x(S) \leq |S| - 1 + y(S)$$
$$\forall S \subset V : 2 \leq |S| \leq |V| - p \tag{5.11}$$

$$x(S', i) + x(S', S) + x(i, S) \geq y_i + y(I) - |I|$$
$$\forall I \subset V \setminus \{i\} : \ |I| = p - 1, \ \forall S \subset V \setminus (I \cup \{i\}) \tag{5.12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{5.6}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V. \tag{5.10}$$

This formulation is an adaptation of the formulation that we proposed in Chapter 2 for the multi-depot routing problem, however, there are two drawbacks to using this model, and in particular constraints (5.12), in practice.

The first drawback is related to the separation of constraints (5.12). The exact separation algorithm that we used to separate the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) in the multi-depot routing problem can be adapted to this case, however, since the set of depots is no longer fixed, then the separation algorithm would no longer be polynomial in time as we would need to fix every possible set $I \cup \{i\}$ such that $|I \cup \{i\}| = p$. The other option would be to find a different separation algorithm that would determine the partition of $V \setminus \{i\}$ into the three sets S', S and I and that would, simultaneously, minimize the left-hand side and maximize the right-hand side of constraints (5.12). In our opinion, it is not clear that the resulting algorithm would be polynomial in time. Observe that if we had to resort to heuristic separation algorithms, then this would defeat the purpose of using this formulation in the space of the x and the y variables since we might as well be using a formulation in the space of the x variables based on the one proposed in Section 5.3. One might argue that other adaptations of path elimination constraints could be used instead of constraints (5.12), however, we believe that any effective set of path elimination constraints relies on the fact that the depots and the clients are clearly identified.

The second drawback arises from a symmetry problem that is a consequence of this modeling approach that selects nodes to be the acting depots of the circuits in that it allows for too many alternative representations of the same solution. For example, a circuit $(i_1, i_2, ..., i_m, i_1)$, with $i_j \in V,\ \forall j \in \{1, \ldots, m\}$, can be represented in m different ways, one for each node as the acting depot. It is not clear how this symmetry problem can be addressed with formulations based on these two sets of variables.

In the next chapter we present a new formulation for the Hamiltonian p-median problem that addresses the two drawbacks identified above, which has a higher linear programming relaxation value than the formulation in the space of the x and the y variables, and which is very effective in practice. More precisely, we will show that the $(\leq p)$ constraints and the $(\geq p)$ constraints of the new formulation imply constraints (5.11) and (5.12), respectively, and that both sets can be separated in polynomial time. In addition, the variables that we use in the new formulation are capable of handling symmetry issues.

Chapter 6

The PQR formulation

Contents

6.1 Introduction

In this chapter we propose a new formulation for the Hamiltonian p-median problem. The formulation is based on the concept of acting depot described earlier and uses three new sets of

variables which essentially incorporate the information of which nodes are acting depots on the arcs. More precisely, we propose three new sets of binary variables which distinguish between the cases of whether or not an arc $(i, j) \in A$ is used in one of the circuits where (i) i is a depot; (ii) j is a depot; and (iii) neither i nor j are depots. In other words, for each arc $(i, j) \in A$ we create a binary variable p_{ij} which indicates whether or not arc (i, j) is used in one of the circuits where i is an acting depot; a binary variable q_{ij} which indicates whether or not arc (i, j) is used in one of the circuits where j is an acting depot; and a binary variable r_{ij} indicating whether or not arc (i, j) is used in one of the circuits where neither i nor j are acting depots.

This chapter is organized in the following way. In Section 6.2 we present the new formulation and in particular show how to model the $(\leq p)$ and the $(\geq p)$ constraints by using the new p, q and r variables. Then, in Section 6.3 we present theoretical results concerning the new formulation, including a comparison to the model in the space of the x and the y variables described in the previous chapter. Finally, in Section 6.4 we present an alternative formulation based on the one presented by Erdoğan et al. (2016) and compare it in theory to the PQR formulation.

6.2 A formulation in the space of the p, q and r variables

In this section we present a new formulation for the Hamiltonian p-median problem based on the three new sets of p, q and r variables which are defined as follows: p_{ij} is a binary variable such that $p_{ij} = 1$ if arc $(i, j) \in A$ is used in one of the circuits and i is an acting depot, and $p_{ij} = 0$ otherwise; q_{ij} is a binary variable such that $q_{ij} = 1$ if arc $(i, j) \in A$ is used in one of the circuits and j is an acting depot, and $q_{ij} = 0$ otherwise; and r_{ij} is a binary variable such that $r_{ij} = 1$ if arc $(i, j) \in A$ is used in one of the circuits and neither i nor j are acting depots, and $r_{ij} = 0$ otherwise.

For simplification we will refer to an arc $(i, j) \in A$ where i in an acting depot as a p-arc, to an arc $(i, j) \in A$ where j in an acting depot as a q-arc, and to an arc $(i, j) \in A$ where neither i nor j are acting depots as an r-arc. None of the three cases consider the situation where the two nodes i and j can be depots at the same time. Therefore, the definition of the three new sets of variables implicitly prevents solutions where two acting depots are directly linked, that is, if an (unfeasible) path between two depots exists then at least one client node is included in that path.

Observe that the new variables are related to the previously defined arc variables x and acting depot variables y. In fact, if an arc $(i, j) \in A$ is used in any circuit, then it must be either a p-arc, a q-arc or an r-arc. Furthermore, if a node $i \in V$ is an acting depot, then surely there must exist a p-arc outgoing node i and a q-arc ingoing node i, and, conversely, if i is not an acting depot, then there cannot exist a p-arc outgoing node i nor a q-arc ingoing node i. More formally, the x and the y variables and the p, q and r variables are related by the following equalities:

$$x_{ij} = p_{ij} + q_{ij} + r_{ij} \qquad \forall (i, j) \in A \tag{6.1}$$

$$y_i = \sum_{j \in V} p_{ij} \qquad \forall i \in V \tag{6.2}$$

$$y_i = \sum_{j \in V} q_{ji} \qquad \forall i \in V. \tag{6.3}$$

In order to obtain a valid generic integer linear programming model for the Hamiltonian p-median problem in the space of the p, q and r variables, we can simply replace the x and the y variables by using the above relationships in the generic model defined in the space of the x and the y variables of Section 5.4. For simplification in the writing of the constraints, however, we will only replace the x variables and leave the y variables as auxiliary variables in the model.

$$\text{Minimize} \sum_{(i,j) \in A} c_{ij} \left(p_{ij} + q_{ij} + r_{ij} \right) \tag{6.4}$$

$$\text{subject to:} \sum_{j \in V} \left(p_{ij} + q_{ij} + r_{ij} \right) = 1 \qquad \forall i \in V \tag{6.5}$$

$$\sum_{j \in V} \left(p_{ji} + q_{ji} + r_{ji} \right) = 1 \qquad \forall i \in V \tag{6.6}$$

$$\sum_{i \in V} y_i = p \tag{5.7}$$

$$y_i = \sum_{j \in V} p_{ij} \qquad \forall i \in V \tag{6.2}$$

$$y_i = \sum_{j \in V} q_{ji} \qquad \forall i \in V \tag{6.3}$$

$$\{(i,j) \in A : p_{ij} = 1 \wedge q_{ij} = 1 \wedge r_{ij} = 1\}$$
$$\text{contains no circuit with zero depots} \tag{6.7}$$

$$\{(i,j) \in A : p_{ij} = 1 \wedge q_{ij} = 1 \wedge r_{ij} = 1\}$$
$$\text{contains no circuit with two or more depots} \tag{6.8}$$

$$p_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{6.9}$$

$$q_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{6.10}$$

$$r_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{6.11}$$

$$y_i \in \{0,1\} \qquad \forall i \in V. \tag{5.10}$$

Observe that by using the relationships (6.2)–(6.3) and the degree constraints (6.5)–(6.6) we can derive the following equalities, which will be useful throughout the remaining exposition:

$$p(V,i) + r(V,i) = 1 - y_i \qquad \forall i \in V \tag{6.12}$$

$$q(i,V) + r(i,V) = 1 - y_i \qquad \forall i \in V. \tag{6.13}$$

Given that this generic model was directly obtained from the one in the space of the x and the y variables, it is clear that a solution of (6.5)–(6.6), (5.7) and (6.2)–(6.3) is comprised of a

number of disjoint circuits and a set of p nodes of V which were chosen as acting depots. Once again, however, a solution may be unfeasible since there is no guarantee that no circuits with only r-arcs exist, that is, circuits with zero acting depots, nor that there is no circuit with two or more acting depots. Both types of unfeasible circuits are prevented by the generic $(\leq p)$ constraints (6.7) and the generic $(\geq p)$ constraints (6.8), respectively. In order to complete the new proposed formulation, which we denote by PQR formulation, we will present two sets of constraints to model both these generic constraints, which are the topics of Section 6.2.1 and Section 6.2.2, respectively.

6.2.1 Modeling the $(\leq p)$ constraints

In this section we show how to model the generic $(\leq p)$ constraints (6.7) of the PQR formulation. Recall that the $(\leq p)$ constraints can be seen as subtour elimination constraints, since they prevent the existence of circuits without any acting depots.

By definition of the p, q and r variables, a client-only circuit is a circuit in which all arcs are r-arcs. Thus, in order to prevent these types of circuits, we can use any set of subtour elimination constraints known from the literature rewritten with the r variables. As we mentioned in Section 2.2, there exist a number of surveys (see, e.g., Öncan et al. 2009, Godinho et al. 2011, Roberti & Toth 2012) which compare different sets of subtour elimination constraints for the traveling salesman problem. With respect to the PQR formulation we use an adaptation of the ones proposed by Dantzig et al. (1954) as follows:

$$r(S) \leq |S| - 1 \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p. \tag{6.14}$$

Constraints (6.14) state that, for any subset of nodes $S \subset V$ such that $2 \leq |S| \leq |V| - p$, there can only exist at most $|S| - 1$ r-arcs linking nodes of S. Note that the upper limit on $|S|$ given by $|V| - p$ is due to the degree constraints (6.5)–(6.6) and the fact that there are no arcs which link two acting depots by definition of the p, q and r variables.

The information provided by the variables of the PQR formulation allows us to lift constraints (6.14), as the following result shows.

Proposition 17. *The following constraints are valid for the PQR formulation:*

$$p(i, S) + y(S \setminus \{i\}) + r(S) \leq |S| - 1 \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.15}$$

$$q(S, i) + y(S \setminus \{i\}) + r(S) \leq |S| - 1 \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S. \tag{6.16}$$

Proof. We start by proving that constraints

$$y(S \setminus \{i\}) + r(S) \leq |S| - 1 \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S, \tag{6.17}$$

are valid for the PQR formulation. Consider a set $S \subset V$ such that $2 \leq |S| \leq |V| - p$ and a node $i \in S$. Observe that from the domain constraints for the y variables (5.10) we have

$y(S \setminus \{i\}) \leq |S| - 1$. Additionally, from the equalities $p(V, i) + r(V, i) = 1 - y_i$ (6.12) and $q(i, V) + r(i, V) = 1 - y_i$ (6.13), we can conclude that if $y_j = 1$ then both $r(j, V)$ and $r(V, j)$ are equal to 0. Thus, suppose that $y(S \setminus \{i\}) = k$ for some k such that $0 \leq k \leq |S| - 1$. In this situation, and given the previous observation, we know that $r(S) \leq |S| - 1 - k$ and, thus, $y(S \setminus \{i\}) + r(S) \leq |S| - 1$, which are precisely constraints (6.17).

We now prove the validity of constraints (6.15) by lifting constraints (6.17) with the term $\alpha p(i, S)$ on the left-hand side, where α is the lifting coefficient. If $p(i, S) = 0$ then constraints (6.15) are clearly valid. Suppose, then, that $p(i, S) = 1$. In this case we know that the arcs incident to i cannot be r-arcs, thus $r(S) \leq |S| - 2$, and that there is a node in $S \setminus \{i\}$, say j, such that $p_{ij} = 1$, that is, j is an acting client, thus, we have $y(S \setminus \{i\}) = k$ for some k such that $0 \leq k \leq |S| - 2$. Therefore, by using a similar reasoning to the one leading to constraints (6.17), we know that $y(S \setminus \{i\}) + r(S) \leq |S| - 2$, hence, $\alpha = |S| - 1 - (|S| - 2) = 1$, which proves the validity of constraints (6.15). A similar lifting procedure using the term $\alpha q(S, i)$ yields (6.16). □

Constraints (6.15) and (6.16) can be further lifted when $|S| > p$, as shown in the following result.

Proposition 18. *The following constraints are valid for the PQR formulation:*

$$y(S) + r(S) \leq |S| - 1 \qquad \forall S \subset V : p < |S| \leq |V| - p. \tag{6.18}$$

Proof. The proof follows from the fact that $y(S) \leq p \leq |S| - 1$, hence, we can apply the same reasoning based on the equalities (6.12) and (6.13) used to prove the validity of constraints (6.17) in Proposition 17. □

By appropriately using the degree constraints (6.5)–(6.6) and the relationships between the y and the p and q variables (6.2)–(6.3), constraints (6.15)–(6.18) can be equivalently written, respectively, as follows:

$$p(V, S) + r(S', S) \geq 1 - p(i, S') \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.19}$$

$$p(V, S) + r(S', S) \geq 1 - q(S', i) \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.20}$$

$$p(V, S) + r(S', S) \geq 1 - y_i \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.21}$$

$$p(V, S) + r(S', S) \geq 1 \qquad \forall S \subset V : p < |S| \leq |V| - p. \tag{6.22}$$

This alternative cut form suggests that these constraints can be separated by resorting to max-flow/min-cut computations in an appropriate auxiliary graph. In Section 7.2.1 we present a separation algorithm which simultaneously finds violated inequalities of these four sets.

Note that constraints (6.19)–(6.20) are valid if $|S| > p$, even though they are dominated by constraints (6.22) in that case. However, constraints (6.22) would not be valid for $|S| \leq p$.

Therefore, by using constraints (6.22) for $|S| > p$ and constraints (6.19)–(6.20) for $|S| \leq p$, we obtain a valid set of $(\leq p)$ constraints for the PQR formulation and, in particular, we guarantee that every constraint (6.21) is always satisfied. This last observation is important since the separation algorithm which we use is only exact with respect to constraints (6.21), as shall be explained. Nevertheless, since constraints (6.21) dominate the original set (6.14), then they also model the generic $(\leq p)$ constraints (6.7).

6.2.2 Modeling the $(\geq p)$ constraints

In this section we show how to model the generic $(\geq p)$ constraints (6.8) of the PQR formulation. Recall that the $(\geq p)$ constraints prevent the existence of circuits with two or more acting depots and, thus, they can be seen as path elimination constraints.

In the first part of this dissertation we discussed several sets of path elimination constraints for the multi-depot routing problem which we may adapt to the context of the PQR formulation. In particular, we presented a newly developed set of path elimination constraints, which are the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14). For now we focus on adapting these constraints for the case $k = 1$, that is, we show how to adapt the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13). Observe that the case $k = 1$ already provides a valid set of path elimination constraints. Later, in Section 6.3.4, we discuss an adaptation of the more general k-MCC inequalities (2.14) for $k \geq 2$.

The following result presents and proves the validity of a set of $(\geq p)$ constraints for the PQR formulation which are based on the 1-MCC inequalities (2.13).

Proposition 19. *The following multi-cut constraints are valid for the PQR formulation and eliminate circuits with two or more acting depots:*

$$q(S', i) + r(S', S) + p(i, S) \geq y_i \qquad \forall i \in V,\ \forall S \subset V \setminus \{i\}. \tag{6.23}$$

Proof. We first prove the validity of constraints (6.23). Consider a node $i \in V$ and a subset $S \subset V \setminus \{i\}$. These constraints are clearly valid if $y_i = 0$. Suppose then that $y_i = 1$ and that $q(S', i) = r(S', S) = p(i, S) = 0$. Since i is an acting depot then, from the relationships between the y and the p and q variables (6.2)–(6.3), we have $q(S, i) = 1 = p(i, S')$. Note that in the circuit of depot i there can be no more acting depots, hence all remaining arcs must be r-arcs. But then it is not possible to complete the circuit for depot i since $r(S', S) = 0$.

To see why these constraints prevent solutions in which there are two or more depots in the same circuit suppose that i is an acting depot, that is, $y_i = 1$, that there exists a node $j \in V \setminus \{i\}$ such that $y_j = 1$, and that i and j are in the same circuit. In this situation there must exist at least one client node in the path from i to j, which follows from the definition of the p, q and r variables, namely that there are no direct links between acting depots. If we consider that the

set of nodes which are in the path from i to j are in S' and that the remaining nodes, except i and j, are in S, we obtain $p(i, S) = 0$, since the arc leaving i goes to S', and $q(S', i) = 0$, since the arc entering i comes from S. In addition, because j is a depot, then there is no r-arc incident to j and, thus, regardless of whether $j \in S$ or $j \in S'$, we have $r(S', S) = 0$. □

The proof of validity for the multi-cut inequalities (6.23) and the 1-MCC inequalities (2.13) is similar, however, with respect to the multi-cut inequalities (6.23) a slight modification is required due to the fact that the other acting depot involved in the unfeasible path other than i is now either in S' or S. Observe that the multi-cut inequalities (6.23) are not a direct adaptation of the 1-MCC inequalities (2.13). In fact, in the case of the multi-cut inequalities (6.23) all nodes of V are part of either S', S or are the node i, whereas in the 1-MCC inequalities (2.13) the other depots other than the one in the multi-cut are not part of the arcs of the multi-cut.

We also observe that the direct adaptation of the 1-MCC inequalities (2.13) presented in Section 5.4.2, namely constraints $x(S', i) + x(S', S) + x(i, S) \geq y_i + y(I) - |I|$ (5.12), was defined for a partition of $V \setminus \{i\}$ into three subsets, whereas the multi-cut inequalities (6.23) are defined for a partition of $V \setminus \{i\}$ into only two subsets. The difference in the number of subsets in the partition is relevant for the complexity of the corresponding separation algorithm and, in fact, we will show in Section 7.2.2 that the multi-cut inequalities (6.23) can be separated in polynomial time by resorting to max-flow/min-cut computations in an adequate graph.

6.3 Theoretical investigations on the PQR formulation

In this section we present a number of theoretical results concerning the PQR formulation. In Section 6.3.1 we present a compact system of inequalities which shares similarities to the ones presented in Section 2.3 for the multi-depot routing problem and which were based on the arc-depot assignment variables z. In particular, we show that it is possible to establish a relationship to the multi-cut inequalities $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.23) of the PQR formulation.

In Section 6.3.2 we present symmetry-breaking constraints for the PQR formulation, including a set of constraints that deal with the symmetries arising in formulations which use the acting depot concept and that result from the relationship established in Section 6.3.1.

Then, in Section 6.3.3 we present a theoretical comparison of the PQR formulation to the model defined in the space of the x and the y variables of Section 5.4 and, finally, in Section 6.3.4 we discuss the adaptation of the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14) for $k \geq 2$ to the context of both the model defined in the space of the x and the y variables and of the PQR formulation.

6.3.1 A compact representation of the $(\geq p)$ constraints

We showed in Section 2.4.2 that the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) of the multi-depot routing problem are equivalent, in terms of the corresponding linear programming relaxation, to a compact system of inequalities based on the arc-depot assignment variables z. In particular, this was shown in Proposition 5 and the proof, which was based on the max-flow/min-cut theorem, allowed us to design an exact separation algorithm for the original 1-MCC inequalities (2.13). In this section we show that a similar result exists with respect to the $(\geq p)$ constraints $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.23) of the PQR formulation. Since this relationship is important, we will present all the important details despite the similarities.

The underlying idea is to observe that the multi-cut inequalities (6.23) can be viewed as cut constraints in a 3-layered graph and, thus, we can derive a corresponding compact system of inequalities which is a network flow model. The set of nodes of the 3-layered graph is composed of three copies of each node of V. These copies are divided into three subsets, which are the three layers of the 3-layered graph, each subset with a copy of each original node. The first layer and the third layer, each composed of $|V|$ nodes, represent the copies of the nodes of the original graph that are viewed as the acting depots. The two layers correspond to viewing them, respectively, as starting points and as ending points of one of the p circuits. The second layer represents the copies of nodes of V that are viewed as the acting client nodes. The arc set of the 3-layered graph is also partitioned into three subsets. The first subset corresponds to the arcs going from the first layer to the second layer, with an arc existing for every existing p variable (i.e., p-arcs). The second subset corresponds to the arcs between the nodes in the second layer, which are represented by the r variables in the original graph (i.e., r-arcs). Finally, the second layer has arcs linking it to the third layer, with an arc existing if and only if a corresponding q variable exists (i.e., q-arcs).

To model the $(\geq p)$ constraints of the PQR formulation in the 3-layered graph we need to guarantee the existence of p paths such that each path starts in a node in the first layer and ends in a node in the third layer, with the additional constraint that the start and the end nodes of a path are copies of the same corresponding node in the original graph. Consider the binary variables $z^k_{ij} = 1$ if an arc (i, j) is in the path from the copy of node k in the first layer to its copy in the third layer, and $z^k_{ij} = 0$ otherwise. The same variables also indicate whether or not arc (i, j) is used in the circuit of the acting depot k in the original graph. The $(\geq p)$ constraints of the PQR formulation may be modeled by the compact system of inequalities below:

$$\sum_{j \in V} z^k_{kj} = y_k \qquad \forall k \in V \tag{6.24}$$

$$\sum_{j \in V} z^k_{jk} = y_k \qquad \forall k \in V \tag{6.25}$$

$$\sum_{j \in V} z^k_{ji} = \sum_{j \in V} z^k_{ij} \qquad \forall k \in V,\ \forall i \in V :\ i \neq k \tag{6.26}$$

$$z^k_{kj} = p_{kj} \qquad \forall k \in V,\ \forall (k,j) \in A \tag{6.27}$$

$$z^k_{jk} = q_{jk} \qquad \forall k \in V,\ \forall (j,k) \in A \tag{6.28}$$

$$z^k_{ij} \leq r_{ij} \qquad \forall k \in V,\ \forall (i,j) \in A :\ i,j \neq k \tag{6.29}$$

$$z^k_{ij} \in \{0,1\} \qquad \forall k \in V,\ \forall (i,j) \in A. \tag{6.30}$$

For each $k \in V$, constraints (6.24)–(6.26) define a network flow system as follows. If k is an acting client, that is, if $y_k = 0$, then there is no flow corresponding to k on any arc. If k is an acting depot, that is, if $y_k = 1$, then constraints (6.24)–(6.25) state that $y_k = 1$ units of flow leave node k through an outgoing arc and enter node k through an ingoing arc, respectively, whereas constraints (6.26) guarantee the flow conservation on each node $i \in V \setminus \{k\}$. By linking the z variables with the p and the q variables through constraints (6.27) and (6.28), respectively, we ensure that the flow for acting depot k can only flow on p-arcs and q-arcs which are incident to k and, thus, we guarantee that the start and the end nodes of each path in the 3-layered graph are the same. Observe that the linking constraints (6.27) and (6.28) are written as equalities instead of inequalities of the less than or equal to form, however, both representations can be shown to be equivalent, similarly to what was proved in Proposition 1 in the context of the multi-depot routing problem.

Proposition 20. *The projection of the linear programming relaxation of the system of inequalities* (6.24)–(6.30) *onto the space of the p, q, r and y variables is given by the multi-cut inequalities* (6.23) *and $p_{ij} \geq 0,\ \forall (i,j) \in A,\ q_{ij} \geq 0,\ \forall (i,j) \in A,\ r_{ij} \geq 0,\ \forall (i,j) \in A$ and $y_i \geq 0,\ \forall i \in V$.*

Proof. This follows from the max-flow/min-cut theorem and the interpretation of the system of inequalities (6.24)–(6.30) in the 3-layered graph. The max-flow/min-cut theorem states that, for each node k, y_k units of flow are sent from its copy in the first layer, say k^1, to its copy in the third layer, say k^3, without passing through its copy in the second layer (due to the flow-conservation constraints (6.26) which are not defined for the copy of node k in the second layer), with arc capacities given by the values of the p, q and r variables, if and only if every cut separating k^1 from k^3 has capacity with value at least y_k. This last requirement corresponds to constraints $q(S',k) + r(S',S) + p(k,S) \geq y_k$, where S' and S form a partition of $V \setminus \{k\}$, which are precisely constraints (6.23). □

The proof of Proposition 20 is essentially an adaptation of the proof of Proposition 5. Nevertheless, it is important to explicitly show it here since it will be fundamental to understand the symmetry-breaking constraints that deal with the symmetries inherent in formulations with acting depots which we will present in the next section. Additionally, Proposition 20 also has

an important consequence in that it will allow us to derive a polynomial-time exact separation algorithm for the multi-cut inequalities (6.23), which we will present in Section 7.2.2, that relies on max-flow/min-cut computations in the 3-layered graph.

6.3.2 Breaking symmetries in the PQR formulation

In this section we discuss two types of symmetries observed in the solutions obtained by the PQR formulation and show how they can be resolved. The reason why we dedicate a section to this topic is two-fold. Firstly, symmetry-breaking in the PQR formulation is not straightforward in what concerns the symmetries arising from the use of the concept of acting depot. Secondly, the computational results will show that symmetry-breaking constraints in the PQR formulation are fundamental for designing an efficient branch-and-cut algorithm.

Symmetry of type I - Reducing the number of candidate acting depots in a circuit

Recall that, as a consequence of the acting depot modeling approach, a circuit of the form $(i_1, i_2, ..., i_m, i_1)$, with $i_j \in V,\ \forall j \in \{1, \ldots, m\}$, can be represented in m different ways, depending on the m possible acting depots. To address this first symmetry problem, we use constraints that are motivated by the well-known idea (see, e.g., Campêlo, Corrêa & Frota 2004) that the acting depot in any circuit $(i_1, i_2, ..., i_m, i_1)$ should be the node with the lowest index to reduce the m possible representations into one.

For example, consider a circuit $(3, 2, 7, 6, 3)$. This circuit admits four different (but equivalent in terms of cost) representations in the PQR formulation depending on whether node 3, node 2, node 7 or node 6 is chosen as the acting depot. However, if we state that the acting depot of a given circuit should be the node with the lowest index, then only node 2 could be the acting depot of circuit $(3, 2, 7, 6, 3)$ and, thus, only one of the four representations would be a possible solution of the PQR formulation.

This idea can be easily implemented by using the z variables of the compact system of inequalities presented in the previous section as follows:

$$z^k_{kj} = 0 \qquad \forall k \in V,\ \forall (k,j) \in A:\ k > j \tag{6.31}$$

$$z^k_{jk} = 0 \qquad \forall k \in V,\ \forall (j,k) \in A:\ k > j \tag{6.32}$$

$$z^k_{ij} = 0 \qquad \forall k \in V,\ \forall (i,j) \in A:\ k > \min\{i,j\}. \tag{6.33}$$

These constraints collectively impose that the depot of each circuit should be the node with the lowest index by disallowing the use of arcs which do not satisfy this condition. For instance, constraints (6.33) state that an arc $(i, j) \in A$ cannot be used in the circuit of an acting depot $k > \min\{i, j\}$ since this would imply that a node with a lower index than k would be an acting client in the circuit of depot k.

The three sets of symmetry-breaking constraints (6.31)–(6.33) can be partially adapted to the PQR formulation by considering the following set of symmetry-breaking constraints:

$$p_{ij} = 0 \qquad \forall (i,j) \in A : i > j \tag{6.34}$$

$$q_{ij} = 0 \qquad \forall (i,j) \in A : i < j. \tag{6.35}$$

Constraints (6.34)–(6.35) prevent some alternative representations of the same circuit, but do not guarantee that the depot of a circuit will be the node with the lowest index. For example, whereas they eliminate the solution $(3, 2, 7, 6, 3)$ since $p_{32} = 0$, they would not eliminate solutions such as the circuit $(3, 7, 2, 6, 3)$.

It is not straightforward how to ensure that the acting depot of any circuit should be the node with the lowest index in the context of the PQR formulation. However, we can indirectly provide such a set of constraints by using the symmetry-breaking constraints (6.31)–(6.33) together with the result of Proposition 20. Recall that the idea of the compact system of inequalities presented in the previous section is to ensure that y_k units of flow are sent from the copy of each node $k \in V$ in the first layer to its copy in the third layer of the 3-layered graph. The result of Proposition 20, which is based on the max-flow/min-cut theorem, states that this condition may be modeled by using multi-cut constraints instead. Thus, if in the proof of Proposition 20 we restrict the arcs in which the flow for node $k \in V$ can pass by removing arcs in the 3-layered graph according to constraints (6.31)–(6.33), this also restricts the arcs allowed in the multi-cut. This leads to the following restricted multi-cut constraints that can be used for symmetry-breaking purposes:

$$q(S', i) + r(S', S) + p(i, S) \geq y_i \qquad \forall i \in V,\ \forall S \subset V \setminus \{1, \ldots, i\}. \tag{6.36}$$

The restricted multi-cut inequalities (6.36) use all the information provided by the symmetry-breaking constraints (6.31)–(6.33) and, therefore, they ensure that the acting depot in any circuit is the node with the lowest index. We formalize the relationship between the compact system of inequalities based on the z variables with the addition of constraints (6.31)–(6.33) and constraints (6.36) in the following result, however, we omit the proof since it is similar to the proof of Proposition 20.

Proposition 21. *The projection of the linear programming relaxation of the system of inequalities* (6.24)–(6.33) *onto the space of the p, q, r and y variables is given by the restricted multi-cut inequalities* (6.36) *and $p_{ij} \geq 0,\ \forall (i,j) \in A,\ q_{ij} \geq 0,\ \forall (i,j) \in A,\ r_{ij} \geq 0,\ \forall (i,j) \in A$ and $y_i \geq 0,\ \forall i \in V$.*

Symmetry of type II - Eliminating reversed circuits

The second type of symmetry arises in symmetric cost instances and is a consequence of a modeling approach based on directed graphs. This issue had already been observed for the multi-depot routing problem and symmetry-breaking constraints were proposed in Section 3.6.4.

Essentially, a circuit and its reverse both have the same cost if arc costs are symmetric, hence, both circuits represent equivalent solutions even if they are structurally different. Observe that for circuits with only two nodes this problem is non-existent assuming that symmetry-breaking constraints of type I are present in the formulation, thus, we focus on circuits which have at least three nodes. These equivalent solutions can pose a non-negligible problem if a large number of the p circuits, say m, in any solution are circuits with at least three nodes, since by combining both possible orientations for these $m \leq p$ circuits we have 2^m different representations of the same solution. In order to eliminate alternative solutions and effectively only allow one single representation instead of 2^m, we can use the following symmetry-breaking constraints in the PQR formulation:

$$\sum_{k \geq j} q_{ki} \geq p_{ij}, \qquad \forall i, j \in V,\ i \neq j. \tag{6.37}$$

Constraints (6.37) state that, for two distinct nodes $i, j \in V$, if i is an acting depot and node j is visited immediately after i, then the node k visited just before i should be such that $k \geq j$. In other words, we enforce that the first node visited after the acting depot is given a lower index than the node visited just before the acting depot. Notice that the case $k = j$ is required in order to avoid eliminating solutions that are composed of two-node circuits.

Furthermore, we can lift constraints (6.37) to:

$$\sum_{k \geq j} q_{ki} \geq \sum_{k \geq j} p_{ik}, \qquad \forall i, j \in V,\ i \neq j. \tag{6.38}$$

To see that this lifted version is valid consider a pair of distinct nodes i and j of V and notice that: (i) the right-hand side of constraints (6.38) is still at most 1 due to the outdegree constraints (6.5); (ii) if $p_{ij} = 1$, then the validity of constraints (6.38) follows from the validity of the original constraints (6.37); (iii) if $p_{ij} = 0$ and $p_{ik'} = 1$ for a node $k' > j$ then the validity of constraints (6.38) follows from the validity of the original constraints (6.37) for the case when $j = k'$; and (iv) for $j = n$ constraints (6.38) and the original constraints (6.37) are the same.

Constraints analogue to constraints (6.38), that is, constraints in which the q variables are on the right-hand side and the p variables are on the left-hand side, can also be defined, however they can be shown to be redundant in the presence of the relationships between the y and the p and q variables (6.2)–(6.3).

We would like to conclude this section by observing that the use of symmetry-breaking constraints in the PQR formulation is fundamental as evidenced by the number of different alternative solutions which represent the same set of circuits. The solutions identified are usually comprised of several circuits with two nodes, thus the second type of symmetry is not as problematic as the first type of symmetry. In the latter case, however, we will provide computational results in Section 7.5.2 that show that the computational effort is substantially reduced by using the restricted multi-cut inequalities (6.36) instead of the original multi-cut inequalities (6.23).

6.3.3 Theoretical comparison of the PQR formulation to the model defined in the space of the x and the y variables

In this section we compare the PQR formulation to the model in the space of the x and the y variables presented in Section 5.4. Recall that the former was created with the purpose of dealing with the drawbacks identified in latter. The first drawback had to do with the separation of the $(\geq p)$ constraints which we believe cannot be solved with a polynomial-time algorithm. The second drawback had to do with the symmetry inherent in formulations which use the concept of acting depot and which is not clear how to handle with the x and the y variables.

In Section 6.3.1 we showed that the PQR formulation addresses the first drawback by establishing a relationship between a compact system of inequalities and the multi-cut inequalities $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.23) of the PQR formulation that will allow us to derive a polynomial-time separation algorithm for the latter. The second drawback was addressed in the previous section, where we presented a set of restricted multi-cut inequalities (6.36) which prevent alternative solutions induced by the use of the acting depot concept.

We now prove that the PQR formulation provides at least the same linear programming relaxation value as the linear programming relaxation value of the model in the space of the x and the y variables by showing that the $(\leq p)$ constraints and $(\geq p)$ constraints of the PQR formulation imply the ones of the model in the space of the x and the y variables, respectively. Observe that this is sufficient since the generic model in the space of the p, q, r and y variables was obtained from the one in the space of the x and y variables by replacing the x variables through the relationships between the x and the p, q and r variables (6.1). For simplification, we provide the ensuing proofs without considering any type of symmetry-breaking constraints.

Proposition 22. *The linear programming relaxation of the PQR formulation implies constraints* $x(S) \leq |S| - 1 + y(S)$ (5.11).

Proof. Consider a set $S \subset V$ such that $2 \leq |S| \leq |V| - p$. By adding $y(S) + p(i, S')$, where $i \in S$, to each side of constraints $p(i, S) + y(S \setminus \{i\}) + r(S) \leq |S| - 1$ (6.15) and by using the relationships between the y and the p variables (6.2) for node i, we obtain

$$y(S) + y(S) + r(S) \leq |S| - 1 + y(S) + p(i, S').$$

Now, by using the relationships between the y and the p variables (6.2) added up for the nodes of S to replace the first $y(S)$ term on the left-hand side, and the relationships between the y and the q variables (6.3) added up for the nodes of S to replace the second $y(S)$ term on the left-hand side, we obtain

$$p(S) + p(S, S') + q(S) + q(S', S) + r(S) \leq |S| - 1 + y(S) + p(i, S').$$

From the relationships between the x and the p, q and r variables (6.1) added up for the arcs between nodes of S, we know that $x(S) = p(S) + r(S) + q(S)$. Hence, we can write the previous inequality as follows:

$$x(S) \leq |S| - 1 + y(S) + p(i, S') - p(S, S') - q(S', S).$$

Finally, the expression $p(i, S') - p(S, S') - q(S', S)$ is non-positive since $i \in S$, hence, the inequality above implies constraint (5.11) for the same set S. A similar proof exist by starting with constraints $q(S, i) + y(S \setminus \{i\}) + r(S) \leq |S| - 1$ (6.16). □

Regarding constraints $x(S', i) + x(S', S) + x(i, S) \geq y_i + y(I) - |I|$ (5.12), we will start by proving that the multi-cut inequalities (6.23) imply the following set of constraints

$$q(S', i) + r(S', S) + p(i, S) \geq y_i + y(I) - |I| \qquad \forall i \in V,\ \forall S \subset V \setminus (I \cup \{i\}) :\ |I| = p - 1, \quad (6.39)$$

which can be seen as constraints (5.12) written with the p, q and r variables instead of the x variables.

Proposition 23. *The linear programming relaxation of the PQR formulation implies constraints* (6.39).

Proof. Consider a constraint (6.23) for a node $i \in V$ and a partition of $V \setminus \{i\}$ into two subsets S' and S, such that S' is partitioned into B' and C' and S is partitioned into B and C, which can be written as:

$$q(B', i) + q(C', i) + r(B', B) + r(C', C) + r(B', C) + r(C', B) + p(i, B) + p(i, C) \geq y_i.$$

Consider now that $I = C \cup C'$ and note that the inequalities $|C| - y(C) \geq r(B', C) + p(i, C)$ and $|C'| - y(C') \geq q(C', i) + r(C', C) + r(C', B)$ are valid since they follow from the equalities $p(V, i) + r(V, i) = 1 - y_i$ (6.12) added up for the nodes of C and $q(i, V) + r(i, V) = 1 - y_i$ (6.13) added up for the nodes of C', respectively. By using these two inequalities to replace the corresponding terms on the left-hand side of the above inequality, we obtain a constraint (6.39) for the partition of $V \setminus \{i\}$ into B', B and I. □

In the proof of Proposition 23, the subsets C and C' correspond to sets of potential acting depots included in S and S', respectively. This shows that the variables of the PQR formulation implicitly hold information about the set I of potential acting depots which appeared in constraints (5.12) and which was artificially introduced in constraints (6.39). Therefore, the following result follows from Proposition 23.

Proposition 24. *The linear programming relaxation of the PQR formulation implies constraints* (5.12).

Proof. It suffices to add adequate non-negativity constraints for the p, q and r variables to the left-hand side of constraints (6.39) and to use the relationships between the x and the p, q and r variables (6.1). □

6.3.4 Generalizations of the multi-cut constraints

In this section we discuss the adaptation of the generalized multi-cut constraints, that is, the k-MCC inequalities $x(S', D') + x(S', S) + x(D', S) \geq |D'|$ (2.14), originally presented for the multi-depot routing problem in Section 2.4.3, both to the PQR formulation and to the model in the space of the x and the y variables. Regarding the latter, recall the set of adapted multi-cut constraints written in their alternative form which we presented in Section 5.4.2, that is,

$$y(I) + x(I, S) + x(S) + x(S, i) + y_i + x(I, i) \leq |S| + 1 + |I| \\ \forall i \in V,\ \forall I \subset V \setminus \{i\} :\ |I| = p - 1,\ \forall S \subset V \setminus (I \cup \{i\})\,. \quad (5.13)$$

By establishing a comparison to the expression of the k-MCC inequalities (2.14) written in their alternative form, namely

$$x(D', S) + x(S) + x(S, D \setminus D') \leq |S| \qquad \forall D' \subset D,\ \forall S \subset C, \quad (2.16)$$

we can clearly see that the adaptation of the k-MCC inequalities (2.14) to the model in the space of the x and the y variables for the Hamiltonian p-median problem involves generalizing the singleton subset $\{i\}$ in constraints (5.13). In order to maintain their overall intuition, in the sense that the equivalent of $I \cup \{i\}$ should represent the set of potential acting depots, we consider two disjoint subsets I_1 and I_2 such that $|I_1 \cup I_2| = p$. Thus, we can derive the following expression:

$$y(I_2) + x(I_2, S) + x(S) + x(S, I_1) + y(I_1) + x(I_2, I_1) \leq |S| + |I_1| + |I_2| \\ \forall I_1, I_2 \subset V :\ I_1 \cap I_2 = \emptyset,\ |I_1| + |I_2| = p,\ \forall S \subset V \setminus (I_1 \cup I_2)\,. \quad (6.40)$$

Observe that by appropriately using the degree constraints (5.2)–(5.3) of the model in the space of the x and y variables, we can write constraints (6.40) in cut form as follows:

$$x(S', I_1) + x(S', S) + x(I_1, S) \geq y(I_1) + y(I_2) - |I_2| \\ \forall I_1, I_2 \subset V :\ I_1 \cap I_2 = \emptyset,\ |I_1| + |I_2| = p,\ \forall S \subset V \setminus (I_1 \cup I_2)\,. \quad (6.41)$$

The above constraints are an adaptation of the k-MCC inequalities (2.14) to the model in the space of the x and y variables, when the nodes of $I_1 \cup I_2$ are acting depots. Note, however, that we are unsure whether in the case of the Hamiltonian p-median problem improvements could be made to the above constraints or not. Recall that the set of nodes is not divided into depot nodes

and client nodes in the Hamiltonian p-median problem in contrast to the multi-depot routing problem and, thus, it is likely that such improvements may exist.

Regarding a generalization of the multi-cut inequalities $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.23) of the PQR formulation, observe that by using a similar reasoning which led to constraints $q(S', i) + r(S', S) + p(i, S) \geq y_i + y(I) - |I|$ (6.39) presented in Section 6.3.3, we can write constraints (6.41) above by replacing the x variables with the appropriate p, q and r variables and obtain:

$$q(S', I_1) + r(S', S) + p(I_1, S) \geq y(I_1) + y(I_2) - |I_2| \\ \forall I_1, I_2 \subset V : \; I_1 \cap I_2 = \emptyset, \; |I_1| + |I_2| = p, \; \forall S \subset V \setminus (I_1 \cup I_2) \, . \tag{6.42}$$

Given the relationship between the multi-cut inequalities (6.23) and constraints (6.39) established in Proposition 23, the earlier indication is that some other generalization of the multi-cut inequalities (6.23) which implies constraints (6.42) should exist. It is still unclear, however, how these other constraints would look like.

We believe that we are not losing much in practice by not using a generalization of the multi-cut inequalities (6.23), since the computational results concerning the multi-depot routing problem which we presented in Section 3.7.3 showed that the use of the k-MCC inequalities (2.14) did not significantly improve the results obtained when compared to the case where only the 1-MCC inequalities $x(S', d) + x(S', S) + x(d, S) \geq 1$ (2.13) were used in the branch-and-cut algorithm. In addition, the separation of a more general set of multi-cut constraints in the PQR formulation also appears to be harder in practice, in the sense that the heuristic separation algorithm which we used in the multi-depot routing problem to separate the k-MCC inequalities (2.14), namely algorithm 3.6, relies on the fact that there exist depots outside of the multi-cut which can then be added to the multi-cut by simply comparing the value of an expression to 1. However, in the multi-cut inequalities (6.23) of the PQR formulation every node of V is either in S', in S or is node i, and, thus, the same reasoning of algorithm 3.6 is not straightforwardly applicable. Observe also that the heuristic separation of the general multi-cut constraints is fundamental since an exact separation algorithm is surely not efficient.

In other words, not only is the expression of a possible generalization of the multi-cut inequalities (6.23) of the PQR formulation unclear, but also it is doubtful that we could make use of it in practice without a more comprehensive study. This is certainly a topic worth of investigation in the future, however, we will not pursue it in this dissertation.

6.4 An alternative formulation

In this section we present an alternative formulation for the Hamiltonian p-median problem, which is an adaptation of the formulation proposed by Erdoğan et al. (2016) for the variant of

the Hamiltonian p-median problem in which two-node circuits are not allowed, and compare it to the PQR formulation. The comparison in this section is theoretical, however, we will also provide some computational results as a complement in Section 7.7.1.

We start by presenting the alternative formulation in Section 6.4.1 and show that it has similarities to some of the models for the multi-depot routing problem presented in Chapter 4. Based on these similarities, we provide a comparison between the alternative formulation and the PQR formulation with respect to the corresponding $(\geq p)$ constraints and $(\leq p)$ constraints in Sections 6.4.2 and 6.4.3, respectively.

6.4.1 The x-v formulation

The formulation proposed by Erdoğan et al. (2016) proved to be effective for the variant of the Hamiltonian p-median problem which does not allow two-node circuits to exist, hence, it is interesting to compare it in theory to the PQR formulation. Observe that the formulation of Erdoğan et al. (2016) is based on undirected graphs and was developed for a variant of the Hamiltonian p-median problem, therefore, a theoretical comparison to the PQR formulation cannot be directly established. Instead, we compare the PQR formulation to an adaptation of the formulation of Erdoğan et al. (2016) to directed graphs, that is, by using variables associated with arcs instead of variables associated with edges, and to the Hamiltonian p-median problem as defined by Branco & Coelho (1990).

The adaptation of the formulation presented by Erdoğan et al. (2016), which we denote by x-v formulation, uses the arc variables x in addition to a set of (acting-)client-(acting-)depot assignment binary variables $v_i^j = 1$ if node $j \in V$ is in the circuit of the acting depot $i \in V$, and $v_i^j = 0$ otherwise. Observe that the variables v_i^i for any $i \in V$ are also defined and, in this case, they equivalently state whether or not i is an acting depot. In other words, we can relate the acting depot variables y to the v variables as follows:

$$y_i = v_i^i \qquad \forall i \in V. \tag{6.43}$$

This relationship shows that the following formulation can be seen as the generic formulation in the space of the x and the y variables presented in Section 5.4 with additional constraints that define the v variables, symmetry-breaking constraints for the v variables, and constraints that model the generic $(\leq p)$ constraints (5.8) and the generic $(\geq p)$ constraints (5.9):

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.1}$$

$$\text{subject to:} \sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{5.2}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in V \tag{5.3}$$

$$\sum_{i \in V} v_i^i = p \tag{6.44}$$

$$v_i^j \leq v_i^i \qquad \forall i, j \in V,\ i \neq j \tag{6.45}$$

$$\sum_{j \in V} v_j^i = 1 \qquad \forall i \in V \tag{6.46}$$

$$v_i^j = 0 \qquad \forall i, j \in V :\ j < i \tag{6.47}$$

$$x(S', S) \geq v_{S'}^i \qquad \forall S \subset V,\ \forall i \in S \tag{6.48}$$

$$v_S^i + x_{ij} \leq v_S^j + 1 \qquad \forall (i,j) \in A,\ \forall S \subset V \setminus \{j\} \tag{6.49}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A \tag{5.6}$$

$$v_i^j \in \{0, 1\} \qquad \forall i, j \in V. \tag{6.50}$$

Constraints (6.44) are equivalent to constraints $\sum_{i \in V} y_i = p$ (5.7) under the relationships (6.43). Constraints (6.45) state that a node $i \in V$ must be an acting depot if a node $j \in V \setminus \{i\}$ is assigned to it. Conversely, no node can be assigned to i if it is not an acting depot. Constraints (6.46) ensure that a node $i \in V$ is either an acting depot or it is assigned to some other acting depot. The symmetry-breaking constraints (6.47) guarantee that the acting depot in every circuit is the node with lowest index. Constraints (6.48) are the $(\leq p)$ constraints of the x-v formulation and state that, for any subset $S \subset V$ and for any $i \in S$, if i is assigned to a node in S' then at least one arc in the cut-set separating S' and S must be used. Finally, constraints (6.49), which are the $(\geq p)$ constraints of the x-v formulation, ensure that if an arc $(i, j) \in A$ is used then the depot to which i is assigned to, which can include itself, must be the same to which j is assigned. Conversely, if i and j are assigned to different depots then arc (i, j) cannot be used.

In contrast to the model presented by Erdoğan et al. (2016), the adapted model presented above uses the arc variables x to allow asymmetric costs to be modeled, whereas the original model uses edge variables. Additionally, normal constraint adaptations that are motivated from changing from a model based on undirected graphs into a model based on directed ones result in constraints (6.48) and (6.49) that are simply directed counterparts of constraints in the original model presented by Erdoğan et al. (2016). Another difference between the two models is, however, more relevant and results from the fact that two-node circuits are not allowed in the variant of the Hamiltonian p-median problem studied in the work by Erdoğan et al. (2016). In order to allow two-node circuits to exist, our adapted model uses single terms "$+x_{ij}$" in constraints (6.49) in contrast to using terms "$+x_{ij} + x_{ji}$" which would provide valid and better constraints in the case where two-node circuits are not allowed.

Observe that the x-v formulation has many similarities to the models for the multi-depot routing problem based on the client-depot assignment variables v discussed in Chapter 4. In particular, the $(\leq p)$ constraints (6.48) are similar to constraints $x(D' \cup S', S) \geq v_{D'}^i$ (4.46) presented in Section 4.3.4. As for the $(\geq p)$ constraints (6.49), they are similar to constraints

$v^i_{D'} + x_{ij} \leq v^j_{D'} + 1$ (4.10) first presented in Section 4.2.4 which generalized the original constraints $v^i_d + x_{ij} \leq v^j_d + 1$ (4.4) for depot subsets. Observe that the generalizations based on arc sets presented in Sections 4.2.2, 4.2.3 or 4.2.5, cannot be straightforwardly adapted to the case of the x-v formulation for the Hamiltonian p-median problem for the same reason why we are required to use "$+x_{ij}$" terms in constraints (6.49) instead of terms "$+x_{ij} + x_{ji}$".

This relationship between the x-v formulation and some of the models presented in Chapter 4 is important since it implies that the work developed in that chapter can be adapted to this section in order to compare the x-v formulation to the PQR formulation, however, given that the set of nodes is no longer divided into depot nodes and client nodes, there are important differences that need to be accounted for. In the remainder of this section we will compare the x-v formulation to the PQR formulation. For simplicity, we ignore symmetry-breaking constraints given that every result is applicable in either case.

In theory we can see that the x-v formulation and the PQR formulation are not comparable in terms of their corresponding linear programming relaxation values. In fact, it is possible to find fractional solutions of the linear programming relaxation of the x-v formulation which do not satisfy the linear programming relaxation of the PQR formulation, and vice-versa. Nevertheless, in order to understand in more detail the differences between the two formulations, more precisely in terms of their corresponding $(\leq p)$ and $(\geq p)$ constraints, we will present in the next two sections some results which will help provide a theoretical link between the x-v and the PQR formulations.

6.4.2 A comparison of the $(\geq p)$ constraints of the x-v formulation and the PQR formulation

One way to establish a link between the $(\geq p)$ constraints of the x-v and of the PQR formulations is to use the compact system of inequalities based on the z variables which we presented in Section 6.3.1. This relationship is very similar to the one presented in Section 4.2.6 for the multi-depot routing problem. Observe that the compact system of inequalities based on the z variables can be strengthened by replacing the linking constraints between the z and the r variables $z^k_{ij} \leq r_{ij}$ (6.29) with the following set of constraints:

$$\sum_{k \in V \setminus \{i,j\}} z^k_{ij} = r_{ij} \qquad \forall (i,j) \in A. \tag{6.51}$$

The reasoning for the validity of these constraints is similar to the one used to derive the 3I^{++} system of Section 2.3.3 for the multi-depot routing problem. In particular, an r-arc $(i,j) \in A$ can only be used in the circuit of exactly one acting depot $k \in V \setminus \{i,j\}$ given that the circuits for each acting depot must be disjoint. Observe that, once again, we could have used inequalities

of the less than or equal to form in constraints (6.51), however, a similar result to the one of Proposition 2 can also be proved in this context.

Another similarity to the multi-depot routing problem case is that the v and the z variables can be related through the equalities

$$v_k^i = \sum_{j \in V} z_{ji}^k \qquad \forall k \in V,\ \forall i \in V, \tag{6.52}$$

or their reversed version, which is equivalent under the flow conservation constraints (6.26),

$$v_k^i = \sum_{j \in V} z_{ij}^k \qquad \forall k \in V,\ \forall i \in V. \tag{6.53}$$

The proof of the following result is, therefore, an adaptation of the proof of the result of Proposition 11 with slight differences motivated by the new context.

Proposition 25. *The linear programming relaxation of the system of inequalities* (6.24)–(6.28)*,* (6.51) *and* (6.30)*, with the addition of either relationships* (6.52) *or* (6.53) *and the domain constraints for the v variables* (6.50)*, implies the $(\geq p)$ constraints $v_S^i + x_{ij} \leq v_S^j + 1$* (6.49) *of the x-v model.*

Proof. Consider an arc $(i, j) \in A$ and a subset $S \subset V \setminus \{j\}$. We will prove the result assuming that $i \in S$. For the case in which $i \notin S$, the proof is similar.

If we add the relationships (6.52) with respect to node j for the nodes in the subset S and weaken the right-hand side of the resulting equation, we obtain:

$$v_S^j = \sum_{d \in S} \sum_{k \in V} z_{kj}^d \geq \sum_{d \in S} z_{ij}^d.$$

Now, if we add $\sum_{d \in S} \sum_{k \in V : k \neq j} z_{ik}^d$ to both sides of the inequality above and use the relationships (6.53) for node i added up for the nodes of S we obtain:

$$v_S^j + \sum_{d \in S} \sum_{k \in V : k \neq j} z_{ik}^d \geq v_S^i.$$

Finally, observe that

$$\sum_{d \in S} \sum_{k \in V : k \neq j} z_{ik}^d = \sum_{k \in V : k \neq j} \sum_{d \in S \setminus \{i\}, d \neq k} z_{ik}^d + \sum_{d \in S \setminus \{i\}} z_{id}^d + \sum_{k \in V : k \neq j} z_{ik}^i,$$

and that

$$\sum_{k \in V : k \neq j} \sum_{d \in S \setminus \{i\}, d \neq k} z_{ik}^d + \sum_{d \in S \setminus \{i\}} z_{id}^d + \sum_{k \in V : k \neq j} z_{ik}^i \leq$$
$$r(i, V \setminus \{j\}) + q(i, S \setminus \{i\}) + p(i, V \setminus \{j\}) \leq x(i, V \setminus \{j\}).$$

In the expression above, the first inequality follows from the linking constraints between the z and the p, q and r variables (6.27), (6.28) and (6.51), respectively, and the last one from the relationships between the x and the p, q and r variables (6.1). Then, by using the outdegree constraints (5.2) for the x variables we obtain

$$v_S^j + 1 \geq v_S^i + x_{ij},$$

which are constraints (6.49) for node j and the subset S. □

The result of Proposition 25 does not allow us to establish any dominance relationship between the x-v formulation and the PQR formulation. However, and given the result of Proposition 20, we can conclude that the $(\geq p)$ constraints of both models are implied by the same system of inequalities based on the z variables. What we will see in practice is that, with respect to the $(\geq p)$ constraints, the linear programming relaxation value of the PQR formulation is higher in most cases than that of the x-v formulation.

6.4.3 A comparison of the $(\leq p)$ constraints of the x-v formulation and the PQR formulation

The link between the x-v formulation and the PQR formulation with respect to their corresponding $(\leq p)$ constraints can be derived from the adaptation of constraints $z^d(\{d\} \cup S) \geq v_d^i$ (4.43) to the context of the Hamiltonian p-median problem. More precisely, we define the following constraints:

$$z^k(S', S) \geq v_k^i \qquad \forall S \subset V,\ \forall k \in S',\ \forall i \in S. \tag{6.54}$$

The validity of these constraints is easy to establish based on similar arguments to those used to derive constraints (4.43) for the multi-depot routing problem in Section 4.3.3. If we add this set of exponentially-many constraints to the system of inequalities based on the z variables, we can prove the two following results.

Proposition 26. *The linear programming relaxation of the system of inequalities* (6.24)–(6.28)*,* (6.51) *and* (6.30)*, with the addition of either relationships* (6.52) *or* (6.53)*, the domain constraints for the v variables* (6.50) *and constraints* (6.54)*, implies the following constraints*

$$\begin{aligned} x(S', S) \geq\ & v_{L_1}^{i_1} + \ldots + v_{L_k}^{i_k} \\ & \forall S \subset V :\ |S| \geq k,\ \forall \{i_1, \ldots, i_k\} \subset S,\ \forall \textit{ partitions } L_1, \ldots, L_k \textit{ of } S', \end{aligned} \tag{6.55}$$

which include as a special case the $(\leq p)$ constraints (6.48) *of the x-v model when $k = 1$.*

Proof. Let $k \geq 1$ and consider $i_1, \ldots, i_k$ distinct nodes of a set $S \subset V$ and $L_1, \ldots, L_k$ a partition of $S' = V \setminus S$. For each $m = 1, \ldots, k$ consider the following constraints (6.54)

$$z^k(S', S) \geq v_k^{i_m} \qquad \forall k \in L_m,$$

which can be equivalently written as

$$\sum_{j \in S'} z^k(j, S) \geq v_k^{i_m} \qquad \forall k \in L_m.$$

By adding all of these constraints for $m = 1, \ldots, k$, we obtain:

$$\sum_{m=1}^{k} \sum_{d \in L_m} \sum_{j \in S'} z^d(j, S) \geq \sum_{m=1}^{k} \sum_{d \in L_m} v_d^{i_m} = v_{L_1}^{i_1} + \ldots + v_{L_k}^{i_k}.$$

The right-hand side of the above constraint is equal to the right-hand side of constraints (6.55). As for the left-hand side, observe that it can be written as:

$$\sum_{m=1}^{k} \sum_{d \in L_m} \sum_{j \in S'} z^d(j, S) = \sum_{d \in S'} \sum_{j \in S'} z^d(j, S) = \sum_{d \in S'} z^d(d, S) + \sum_{j \in S'} \sum_{d \in S', d \neq j} z^d(j, S). \tag{6.56}$$

Thus, by using the linking constraints between the z and the p variables (6.27) for the first term of the above rightmost expression, the linking constraints (6.51) for the second term of the above rightmost expression and, lastly, the relationships between the x and the p, q and r variables (6.1) we obtain

$$\sum_{d \in S'} z^d(d, S) + \sum_{j \in S'} \sum_{d \in S', d \neq j} z^d(j, S) \leq p(S', S) + r(S', S) \leq x(S', S),$$

which completes the proof. □

Proposition 27. *The linear programming relaxation of the system of inequalities* (6.24)–(6.28)*,* (6.51) *and* (6.30)*, with the addition of either relationships* (6.52) *or* (6.53)*, the domain constraints for the* v *variables* (6.50) *and constraints* (6.54)*, implies lifted* $(\leq p)$ *constraints* $p(i, S) + y(S \setminus \{i\}) + r(S) \leq |S| - 1$ (6.15) *and* $q(S, i) + y(S \setminus \{i\}) + r(S) \leq |S| - 1$ (6.16)*.*

Proof. Consider a set $S \subset V$ such that $2 \leq |S| \leq |V| - p$, a node $i \in S$ and, for each $d \in S \setminus \{i\}$, an inequality (6.54) as follows:

$$z^d(S' \cup \{d\}, S \setminus \{d\}) \geq v_d^i.$$

By adding these constraints for all $d \in S \setminus \{i\}$ we obtain:

$$\sum_{d \in S \setminus \{i\}} z^d(S' \cup \{d\}, S \setminus \{d\}) \geq v_{S \setminus \{i\}}^i.$$

Now consider the rightmost expression under (6.56) in the proof of Proposition 26 in the case in which $k = 1$. If we add that expression to the above inequality we can derive:

$$\sum_{d \in S'} z^d(d, S) + \sum_{j \in S'} \sum_{d \in S', d \neq j} z^d(j, S) + \sum_{d \in S \setminus \{i\}} z^d(S' \cup \{d\}, S \setminus \{d\}) \geq v_{S'}^i + v_{S \setminus \{i\}}^i = 1 - y_i.$$

By manipulating the expression on the left-hand side of the above inequality, the details of which we omit for simplification, we can arrive at the following inequality:

$$\sum_{d \in V \setminus \{i\}} z^d(d, S) + \sum_{j \in S'} \sum_{d \in V \setminus \{i\}, d \neq j} z^d(j, S \setminus \{d\}) \geq 1 - y_i.$$

If we add $z^i(i, S)$ to both sides, we obtain:

$$\sum_{d \in V} z^d(d, S) + \sum_{j \in S'} \sum_{d \in V \setminus \{i\}, d \neq j} z^d(j, S \setminus \{d\}) \geq 1 - z^i(i, S').$$

Thus, we can derive,

$$1 - p(i, S') \leq 1 - z^i(i, S') \leq \sum_{d \in V} z^d(d, S) + \sum_{j \in S'} \sum_{d \in V \setminus \{i\}, d \neq j} z^d(j, S \setminus \{d\}) \leq p(V, S) + r(S', S),$$

which are exactly constraints (6.15) in their cut form.

Regarding constraints (6.16), we can use the flow-conservation constraints (6.26) to derive a set of constraints which are the reverse of constraints (6.54), namely

$$z^k(S, S') \geq v^i_k \qquad \forall S \subset V,\ \forall k \in S',\ \forall i \in S, \tag{6.57}$$

and use a similar reasoning. □

Once again, these results do not allow us to establish a dominance relationship between the x-v formulation and the PQR formulation. However, in practice, we will see that the linear programming relaxation value of the x-v formulation is higher in most cases than that of the PQR formulation, with respect to the $(\leq p)$ constraints.

6.5 Concluding remarks

In this chapter we proposed a new formulation for the Hamiltonian p-median problem based on the concept of acting depot presented in Chapter 5. The new formulation uses a novel idea by incorporating the information of which nodes are acting depots on the arcs. More precisely, we defined three sets of binary variables, the p, q and r variables, such that p_{ij} indicates whether or not an arc $(i, j) \in A$ is used and i is an acting depot, q_{ij} indicates whether or not an arc (i, j) is used and j is an acting depot, and r_{ij} indicates whether or not an arc (i, j) is used and neither i nor j are acting depots.

The new formulation, which we denoted by PQR formulation, allowed us to address the two drawbacks identified in the formulation in the space of the x and the y variables presented in Section 5.4, which were the complexity of the separation algorithm for the $(\geq p)$ constraints and the symmetries induced by the use of the acting depot concept. The former drawback was addressed in the PQR formulation by presenting a set of $(\geq p)$ constraints, namely an adaptation

of the multi-cut constraints presented in Chapter 2 for the multi-depot routing problem, which can be separated in polynomial time. We also showed that the $(\geq p)$ constraints of the PQR formulation could be improved to act as symmetry-breaking constraints that prevent alternative solutions which only differ on the acting depot chosen.

Furthermore, we proved that the linear programming relaxation of the PQR formulation is at least as good as the linear programming relaxation of the formulation in the space of the x and the y variables and, additionally, we compared the PQR formulation to an alternative formulation which is an adaptation of a formulation from the literature that was originally proposed for the variant of the Hamiltonian p-median problem in which two-node circuits are not allowed.

In the next chapter we present a branch-and-cut algorithm based on the PQR formulation to solve the Hamiltonian p-median problem and additional computational results, including a comparison, in practice, of the PQR formulation and the alternative formulation.

Chapter 7

A branch-and-cut algorithm

Contents

7.1 Introduction

In this chapter we present a branch-and-cut algorithm based on the PQR formulation presented in Chapter 6 and a set of computational results to assess its performance, as well as complementary computational results with a number of different objectives. We advise reading Sections 3.1 and 3.2 and the introduction of Section 3.3 to refresh the concepts related to branch-and-cut algorithms and separation algorithms, respectively.

This chapter is organized in the following way. In Section 7.2 we present separation algorithms relevant to the constraints of the PQR formulation. In Section 7.3 we provide information on the test instances used in the computational experiments. In Section 7.4 we present the outline of the branch-and-cut algorithm. In Section 7.5 we present results to show the importance, in practice, of symmetry-breaking constraints and of the liftings of the $(\leq p)$ constraints of the PQR formulation discussed in Section 6.2.1. In Section 7.6 we present the results of the branch-and-cut algorithm for the test instances. In Section 7.7 we present complementary computational results to compare the PQR formulation to the alternative x-v formulation presented in Section 6.4.1, and to assess the performance of an adaptation of the branch-and-cut algorithm to solve the variant of the Hamiltonian p-median problem in which two-node circuits are not allowed. Finally, we finish with some concluding remarks in Section 7.8.

7.2 Separation algorithms

In this section we present separation algorithms for the $(\leq p)$ and the $(\geq p)$ constraints of the PQR formulation. We will use two different auxiliary graphs in the separation algorithms, however, given the similarities to the graphs depicted in Figures 3.1 and 3.2 with respect to the multi-depot routing problem, we omit some details for the sake of brevity.

The first auxiliary graph is the 3-layered graph described in Section 6.3.1. The second auxiliary graph is a 2-layered graph which is essentially the 3-layered graph in which the third layer is removed, that is, we only consider the two layers which correspond to viewing the nodes as acting depots which represent the starting point of one of the p circuits and as acting clients, respectively. Additionally, we add a node s with an outgoing arc to every node in the first layer.

We present the separations algorithms for the $(\leq p)$ constraints and the $(\geq p)$ constraints of the PQR formulation in Sections 7.2.1 and 7.2.2, respectively.

7.2.1 Separation of the $(\leq p)$ constraints of the PQR formulation

In order to separate the $(\leq p)$ constraints of the PQR formulation, namely

$$p(V,S) + r(S',S) \geq 1 - p(i,S') \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.19}$$

Algorithm 7.1

Require: A point (p^*, q^*, r^*, y^*) and the auxiliary 2-layered graph.

for all $i \in V$ **do**
  Set the capacities of the arcs from s to every node in the first layer to 1, the capacities of the arcs from the nodes in the first layer to the nodes in the second layer to the corresponding value p^*, the capacities of the arcs between nodes in the second layer to the corresponding value r^*, and determine the maximum flow w from s to the copy of node i in the second layer.
  if $w < 1$ **then**
    Given the corresponding minimum cut, set S as the subset of nodes for which the corresponding copies in the second layer are in the same shore as the copy of node i in the second layer.
    if $|S| > p$ **then**
      S defines a violated inequality (6.22).
    else
      Evaluate $w < 1 - p^*(i, S')$ and $w < 1 - q^*(S', i)$. If at least one expression is true, then S defines a violated inequality (6.19) or (6.20) depending on which expression was evaluated as true. If both were, then add only the inequality for which the violation is the greatest.
    end if
  end if
end for

Algorithm 7.2

Require: A point (p^*, q^*, r^*, y^*).

Find the connected components induced by (p^*, q^*, r^*, y^*) by considering all arcs $(i, j) \in A$ such that $p^*_{ij} > 0$ or $q^*_{ij} > 0$ or $r^*_{ij} > 0$ (e.g., by using a depth-first search algorithm).
for all connected components where every node of the component is such that $y^* = 0$ **do**
  The set S comprised of the nodes of the component defines a violated inequality (6.19), (6.20) and (6.22). Add the violated inequality for which the violation is the greatest.
end for

$$p(V, S) + r(S', S) \geq 1 - q(S', i) \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.20}$$

$$p(V, S) + r(S', S) \geq 1 - y_i \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.21}$$

$$p(V, S) + r(S', S) \geq 1 \qquad \forall S \subset V : p < |S| \leq |V| - p, \tag{6.22}$$

we provide two algorithms.

Algorithm 7.1 is based on max-flow/min-cut computations. Intuitively, observe that the left-hand side of any of the above constraints corresponds to a cut in the 2-layered graph in which all nodes of the first layer are on the same shore given the common term $p(V, S)$. Thus, the min-cuts obtained in algorithm 7.1 are cuts that partition the nodes in the second layer into two subsets which originate two corresponding subsets of the original nodes of V, namely S' and S in the above expressions. Then, depending on the cardinality of S, either violated constraints

Algorithm 7.3

Require: A point (p^*, q^*, r^*, y^*) and the auxiliary 3-layered graph.

1: **for all** $i \in V$ such that $y_i^* > 0$ **do**
2: Temporarily remove the copy of node i from the second layer. Set the capacities of the arcs from the nodes in the first layer to the nodes in the second layer to the corresponding value p^*, the capacities of the arcs between nodes in the second layer to the corresponding value r^*, the capacities of the arcs from the nodes in the second layer to the nodes in the third layer to the corresponding value q^*, and determine the maximum flow w from the copy of node i in the first layer to the copy of node i in the third layer.
3: **if** $w < y_i^*$ **then**
4: The corresponding minimum cut defines a violated inequality (6.23) for i in which S is the subset of nodes for which the corresponding copies in the second layer are in the same shore as the copy of node i in the third layer.
5: **end if**
6: **end for**

(6.19), (6.20) or (6.22) are found.

Algorithm 7.1 is not an exact separation algorithm for any of the three sets of constraints (6.19), (6.20) or (6.22). Regarding constraints (6.19) and (6.20), note that the right-hand side depends on the set S'. More precisely, we do not take into account the values $p^*(i, S')$ and $q^*(S', i)$ when determining the min-cut. As for constraints (6.22), the additional condition that $|S| > p$ is also not taken into account during the min-cut computation. Nevertheless, observe that algorithm 7.1 is exact with respect to constraints (6.21). In fact, when algorithm 7.1 stops, for sets S such that $|S| > p$ we know that the value of the min-cut, say w, is such that $w \geq 1 \geq 1 - y_i^*$ for any $i \in S$, and if $|S| \leq p$ we know that $w \geq \max\{1 - p^*(i, S'), 1 - q^*(S', i)\} \geq 1 - y_i^*$ for any $i \in S$. Thus, it is ensured that no inequality (6.21) is violated, and, therefore, no client-only circuits will exist in the solution.

As for algorithm 7.2, it is a heuristic separation algorithm for fractional points and an exact separation algorithm for integer points based on the computation of connected components which is very similar to algorithm 3.2 proposed in the first part of this dissertation.

7.2.2 Separation of the $(\geq p)$ constraints of the PQR formulation

In order to separate the multi-cut inequalities $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.23) we provide two algorithms: algorithm 7.3, which is a polynomial-time exact separation algorithm, and algorithm 7.4, which is a polynomial-time exact separation algorithm for integer points and a heuristic separation algorithm for fractional points.

Similarly to the case of the multi-depot routing problem, the multi-cut inequalities (6.23) are cuts in the 3-layered graph. Therefore, algorithm 7.3 is based on max-flow/min-cut computa-

Algorithm 7.4

Require: A point (p^*, q^*, r^*, y^*).

1: Find the connected components induced by (p^*, q^*, r^*, y^*) by considering all arcs $(i, j) \in A$ such that $p^*_{ij} > 0$ or $q^*_{ij} > 0$ or $r^*_{ij} > 0$ (e.g., using a depth-first search algorithm).
2: **for all** connected components where at least two nodes of the component are such that $y^* > 0$ **do**
3: Find a path between two such nodes in the connected component, say a path from i to j, and set S as the complementary set in $V \setminus \{i, j\}$ of the set of nodes in the path found.
4: **if** (p^*, q^*, r^*, y^*) is integer **then**
5: S defines a violated inequality (6.23) for node i.
6: **else**
7: If $q^*(S' \cup \{j\}, i) + r^*(S' \cup \{j\}, S) + p^*(i, S) < 1 - y^*_i$, then S defines a violated inequality (6.23) for node i.
8: **end if**
9: **end for**

tions in the 3-layered graph, with capacities given by the point being separated, and it checks if the max-flow from the copy of a given node $i \in V$ in the first layer to its copy in the third layer is at least y^*_i. If for any node this max-flow value is below y^*_i, then a violated inequality exists. Note that if $y^*_i = 0$ it is not necessary to compute the max-flow.

Algorithm 7.4 is a heuristic separation algorithm for fractional points and an exact separation algorithm for integer points based on the computation of connected components which is very similar to algorithm 3.4 proposed in the first part of this dissertation.

Finally, in order to separate the restricted symmetry-breaking version of the multi-cut inequalities (6.36), we use similar algorithms with only a minor difference in the case of algorithm 7.3, namely in step 2 we not only temporarily remove from the second layer the copy of node i, but also remove the copy of every node j such that $j < i$.

7.3 Test instances and software/hardware configurations

The remainder of this chapter will focus on computational experiments. For these experiments we use two sets, A and B, of instances. The first set A is a subset of the very well-known TSPLIB symmetric traveling salesman problem benchmark instances, namely dantzig42, swiss42, att48, gr48, hk48, eil51, berlin52, brazil58, st70, eil76, pr76, gr96, rat99, kroA100, kroB100, kroC100, kroD100, kroE100 and rd100. The number of nodes in these instances varies from 42 to 100. The set B of instances includes TSPLIB asymmetric traveling salesman problem benchmark instances, namely ftv33 to ftv170, p43, ry48p, ft53, ft70 and kro124p, with a number of nodes which varies from 34 to 171. The complete description for the sets A and B of instances is available at `https://www.iwr.uni-heidelberg.de/groups/comopt/`

`software/TSPLIB95/`. In the symmetric instances of set A where node coordinates are provided instead of explicit cost values, we determine the cost of an arc (i, j), and consequently of arc (j, i), by rounding up the Euclidean distance between i and j.

As in the first part of this dissertation, the computational experiments were conducted on a single thread of an Intel Core i7-4790 3.6GHz processor in a personal computer with 8GB of RAM and within which CPLEX 12.6.1 Concert Technology for C++ was used. All of the code is original, except for the max-flow algorithm which is based on the push-relabel algorithm by Goldberg & Tarjan (1988).

7.4 The outline of the branch-and-cut algorithm

The branch-and-cut algorithm based on the PQR formulation uses the branch-and-cut algorithm framework of CPLEX version 12.6.1 by IBM (2014) through its Concert Technology for C++. Recall that the advantage of using this framework is that the branch-and-cut algorithm is managed by the solver but the user can intervene in specific parts of the algorithm by using callback functions. For an efficient implementation of a branch-and-cut algorithm, the use of callback functions is important, however, since the code in these functions is managed by the user, it is important to use certain techniques to ensure the efficiency of the branch-and-cut algorithm.

The techniques we use in the branch-and-cut algorithm for the Hamiltonian p-median problem are essentially the same as in the branch-and-cut algorithm proposed in Section 3.6 for the multi-depot routing problem. For instance, we already presented in the previous section heuristic separation algorithms for the exponentially-sized sets of constraints involved, as well as symmetry-breaking constraints in Section 6.3.2. We now discuss two other important aspects, namely parameters to control the number of violated inequalities added in each iteration of the cutting plane algorithm in Section 7.4.1, as well as a primal heuristic that finds feasible solutions by using information from the nodes of the branch-and-bound tree in Section 7.4.2.

7.4.1 Parameters for lazy constraint/user cut callback functions

In Section 3.6.2 we explained the usefulness of parameters to control the addition of violated inequalities in lazy constraint/user cut callback functions. In the branch-and-cut algorithm based on the PQR formulation we use the same type of parameters, namely a separation priority list, a parameter to control the number of lazy constraints or user cuts added in total before re-optimizing, and a parameter to control whether or not we should skip some of the separation algorithms in the separation priority list in a given cutting plane iteration. These last two parameters work exactly the same as explained in Section 3.6.2.

The separation priority list in this case depends on the number of circuits in the relaxation of

the Hamiltonian p-median problem in which we do not add any $(\leq p)$ nor $(\geq p)$ constraints, that is, the so-called assignment relaxation. More precisely, consider the problem comprised solely of the degree constraints (5.2)–(5.3) and the domain constraints for the x variables (5.6) with the arc cost minimization objective (5.1). As we mentioned in Section 5.3, this corresponds to the assignment problem. The first step in the branch-and-cut algorithm is to determine the number of circuits in the assignment relaxation, which we denote by p'. Observe that performing this first step is reasonable since the assignment problem can be solved in polynomial time (see, e.g., Wolsey 1998).

The reasoning for performing this first step is based on preliminary computational experiments. Apart from a very small set of test instances, the models described in this dissertation for the Hamiltonian p-median problem, including the PQR formulation, have two properties: (i) for a Hamiltonian p-median problem instance where $p < p'$, the models described in this dissertation provide the same optimal solution and linear programming relaxation value if we do not add any $(\geq p)$ constraints when compared to a complete model; and (ii) for a Hamiltonian p-median problem instance where $p > p'$, the models described in this dissertation provide the same optimal solution and linear programming relaxation value if we do not add any $(\leq p)$ constraints when compared to a complete model. Additionally, we found that when using a complete model, that is, including both $(\leq p)$ and $(\geq p)$ constraints, in a branch-and-cut algorithm, either none or very few violated constraints of the type $(\geq p)$ are identified when $p < p'$, and none or very few violated constraints of the type $(\leq p)$ are identified if $p > p'$.

Based on the above properties, we define the separation priority lists such that the inequalities which are much more likely to be violated are separated first. This translates to the following. For lazy constraints, we define the separation priority list for $p < p'$ as algorithm 7.2 followed by 7.4, whereas for $p > p'$ the order is reversed. With respect to user cuts, and for $p < p'$, the order is algorithm 7.2, algorithm 7.4, algorithm 7.1 and, finally, algorithm 7.3. As for the case $p > p'$, the order is 7.4, algorithm 7.2, algorithm 7.3 and, lastly, algorithm 7.1.

7.4.2 A primal heuristic

Similarly to what was presented in Section 3.6.3 for the multi-depot routing problem, we regularly apply a heuristic procedure to find feasible solutions based on fractional solutions that can hopefully improve the current incumbent or to improve incumbents found during the regular branch-and-bound process.

The heuristic procedure used in the branch-and-cut algorithm based on the PQR formulation is an adaptation of the one described in Section 3.6.3 and it works as follows. Given a non-negative cost function c', we start by finding the cheapest p arcs to start the p circuits. Then, by using a nearest neighbor type of criterion we insert the remaining nodes in the best possible circuit until we form p circuits that cover V. To further optimize these p circuits we apply local

search operators that first swap nodes inside the circuits, then move nodes from their circuit to a different circuit and finally swap nodes inside the circuits again. If the heuristic is applied by using the original cost function c, the solutions obtained will usually be of poor quality, as was the case in the heuristic procedure proposed in Section 3.6.3. Instead, we use the fractional values p^*, q^* and r^* at a given node of the branch-and-bound tree to modify the cost of an arc (i, j) to $c_{ij} \times (1 - p^*_{ij} - r^*_{ij} - q^*_{ij})$. Recall that the reasoning for this modified cost function is that arcs for which the linear programming relaxation value is close to 1 will have a lower cost and, thus, have a higher probability of being chosen in the constructive part of the heuristic.

For nodes of the branch-and-bound tree in which an integer solution was found, the heuristic simply applies the local search operators to the solution found, since it does not require the construction phase. The heuristic is applied at every five nodes during the first 250 nodes explored in the branch-and-bound tree, following which the frequency is decreased to every 10 nodes.

7.5 Preliminary computational experiments

Before we discuss the final results obtained by the branch-and-cut algorithm, we present in this section some preliminary computational experiments that provide insight on the benefits of two aspects of the PQR formulation.

In Section 7.5.1 we show the benefit of separating the lifted versions of the $(\leq p)$ constraints presented in Section 6.2.1. Then, in Section 7.5.2 we present some results that compare the branch-and-cut algorithm to another one in which symmetry-breaking constraints are not used.

7.5.1 Evaluating the effectiveness of using the lifted $(\leq p)$ constraints

In Section 6.2.1 we discussed the $(\leq p)$ constraints of the PQR formulation. In particular, we presented four different sets of constraints, which we recall are as follows written in cut form

$$p(V, S) + r(S', S) \geq 1 - p(i, S') \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.19}$$

$$p(V, S) + r(S', S) \geq 1 - q(S', i) \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.20}$$

$$p(V, S) + r(S', S) \geq 1 - y_i \qquad \forall S \subset V : 2 \leq |S| \leq |V| - p,\ \forall i \in S \tag{6.21}$$

$$p(V, S) + r(S', S) \geq 1 \qquad \forall S \subset V : p < |S| \leq |V| - p, \tag{6.22}$$

and provided a theoretical comparison. Recall that, for sets S such that $|S| > p$, constraints (6.22) imply the other three sets. Additionally, for $|S| \leq p$, constraints (6.19) and (6.20) imply constraints (6.21).

In this section we present some results that show the actual benefit for the branch-and-cut algorithm of separating the lifted $(\leq p)$ constraints. More precisely, we compare three branch-and-cut algorithms. The first branch-and-cut algorithm, which we denote by B&C$_1$, is similar

Table 7.1: Comparison the solution times of different $(\leq p)$ constraints

				B&C$_1$	B&C$_2$	B&C$_3$
Name	p'	p	OPT	t (s)	t (s)	t (s)
kro124p	32	5	35435	37	9	13
		10	35010	9	23	15
		15	34799	817	403	68
		20	34433	187	53	62
		25	34267	455	241	87
		30	34002	35	1	1
		35	34050	0	0	0
		40	34310	0	0	0
		45	35331	1	1	1
		50	37541	1	1	1
rat99	45	5	1237	19	17	8
		10	1212	374	47	68
		15	1195	179	137	39
		20	1184	2279	556	172
		25	1170	1858	342	177
		30	1159	1213	288	41
		35	1153	799	486	50
		40	1145	172	23	15
		45	1142	1	0	0

to the branch-and-cut algorithm presented in Section 7.4 but where only the weaker constraints (6.21) are separated. The second branch-and-cut algorithm, denoted by B&C$_2$, is similar to the branch-and-cut algorithm presented in Section 7.4 in which constraints (6.19) and (6.20) are separated. Finally, the third branch-and-cut algorithm, denoted by B&C$_3$, is the branch-and-cut algorithm presented in Section 7.4.

Table 7.1 shows the comparison results for the asymmetric instance kro124p and the symmetric instance rat99 with the following format. The first four columns indicate the name of the instance, the number p' of circuits in the assignment relaxation, the value of p used, and the optimal value of the instance (OPT), respectively. The remaining three columns show the time taken (t) in seconds to obtain the optimal value for each of the three branch-and-cut algorithms.

The results are self-explanatory and essentially show that the use of the lifted versions of the $(\leq p)$ constraints of the PQR formulation provide significant improvements in terms of the average computational time required to obtain the optimal solution. In particular, observe that by using the standard branch-and-cut algorithm, namely the branch-and-cut algorithm B&C$_3$, we obtain the lowest average computational times. Additionally, the branch-and-cut algorithm

B&C$_2$ also improves the results of the branch-and-cut algorithm B&C$_1$.

7.5.2 Evaluating the effectiveness of using symmetry-breaking constraints

The computational results presented in this section aim to show the benefits of using symmetry-breaking constraints in the PQR formulation. Recall that in Section 6.3.2 we discussed two types of symmetries identified in the solutions of the PQR formulation. The symmetries of type I are related to alternative solutions which only differ on the acting depots chosen for each circuit and are a consequence of the acting depot concept, whereas the symmetries of type II are due to the use of directed graph based formulations for which both orientations of a circuit have the same cost in symmetric cost instances.

The symmetries of type II had already been identified in the multi-depot routing problem and we showed that they can be resolved through the use of the symmetry-breaking constraints

$$\sum_{k \geq j} q_{ki} \geq \sum_{k \geq j} p_{ik}, \qquad \forall i, j \in V,\ i \neq j. \tag{6.38}$$

We will not provide comparative results for these constraints, however, computational testing shows that, similarly to what was observed in Section 3.6.4 with respect to the multi-depot routing problem, the improvements they provide are not substantial, which may once again be attributed to the fact that most solutions are comprised of many circuits with only two nodes. Nevertheless, a slight average decrease in the solution times was observed, hence their use in the branch-and-cut algorithm for symmetric cost instances.

We now present test results to show the effect of the symmetry-breaking constraints of type I. More precisely, we compare the standard branch-and-cut algorithm described previously, in which the symmetry-breaking restricted multi-cut inequalities $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.36) are used, to another similar branch-and-cut algorithm in which the only difference is that the regular multi-cut inequalities (6.23) are used instead. Observe, however, that there is no advantage in using the regular multi-cut inequalities (6.23) over the restricted multi-cut inequalities (6.36) in a branch-and-cut algorithm, unless we explicitly want to allow alternative solutions. In fact, the latter dominate the former and, additionally, they are faster to separate since the max-flow/min-cut computations are performed in smaller-sized auxiliary graphs.

Tables 7.2 and 7.3 show the comparison results, with Table 7.2 focusing on the asymmetric instance ftv170 and Table 7.3 on the symmetric instance rat99. Both tables have the following format. The first three columns indicate the name of the instance, the number p' of circuits in the assignment relaxation and the value of p used, respectively. The remaining six columns are divided into two parts, each with three columns corresponding to, respectively, the linear programming relaxation value (LP), the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit of 10800 seconds was reached (OPT) and the respective time taken (t) in seconds. The first part corresponds to the alternative branch-and-cut

Table 7.2: Evaluating the effectiveness of using symmetry-breaking constraints of type I (1 of 2)

			B&C$_1$			B&C$_2$		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
ftv170	17	5	2662	2683	15	2662	2683	14
		10	2635	2636	8	2635	2636	8
		15	2631	2631	0	2631	2631	0
		20	2631	2631	0	2631	2631	0
		25	2631	2639	972	2635.14	2639	28
		30	2632.95	[2646, 2658]	10800*	2644.38	2658	44
		35	2643.57	[2664, 2705]	10800*	2671.92	2704	290
		40	2670.93	[2713, 2736]	10800*	2714.81	2736	508
		45	2719.42	[2779, 2799]	10800*	2772.13	2799	356
		50	2781.78	[2842, 2884]	10800*	2844.78	2884	735
		55	2861.38	[2937, 3012]	10800*	2954.38	3008	2470
		60	2989.2	[3080, 3212]	10800*	3105.34	3205	3648
		65	3198.03	[3246, 3695]	10800*	3325.09	[3428, 3432]	10800*
		70	3450.56	[3518, 3826]	10800*	3567.63	[3684, 3706]	10800*
		75	**	**	**	3727.48	[3767, 4091]	10800*
		80	**	**	**	3962.44	[4099, 4403]	10800*
		85	**	**	**	4644.5	4777	7647

*Not solved to optimality within the time limit of three hours

** Ran out of memory before the time limit

algorithm, denoted by B&C$_1$, which uses the regular multi-cut inequalities (6.23), and the second part corresponds to the standard branch-and-cut algorithm, denoted by B&C$_2$, which uses the restricted multi-cut inequalities (6.36).

The results for the asymmetric instance ftv170 show a substantial improvement by the use of the restricted multi-cut inequalities (6.36) instead of the regular multi-cut inequalities (6.23). In fact, the branch-and-cut algorithm B&C$_2$ was able to optimally solve eight more instances than the branch-and-cut algorithm B&C$_1$, namely for $p \in \{30, 35, 40, 45, 50, 55, 60, 85\}$. Additionally, a reduction in computational times is observed for instances that were already solved to optimality, such as the instance with $p = 25$ for which the computational time decreased from about fifteen minutes to only 28 seconds. Observe that the results also show that the linear programming relaxation values are, in most cases, substantially higher when we use the restricted multi-cut inequalities (6.36) in the PQR formulation, which also helps to explain the overall improved results.

As for the symmetric instance rat99, the use of the restricted multi-cut inequalities (6.36) instead of the regular multi-cut inequalities (6.23) does not have such a significant effect compared

Table 7.3: Evaluating the effectiveness of using symmetry-breaking constraints of type I (2 of 2)

			B&C$_1$			B&C$_2$		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
rat99	45	5	1228.5	1237	26	1228.5	1237	8
		10	1196.5	1212	62	1196.5	1212	68
		15	1178.17	1195	68	1178.17	1195	39
		20	1164.17	1184	433	1164.17	1184	172
		25	1154.38	1170	83	1154.38	1170	177
		30	1148	1159	508	1148	1159	41
		35	1144.44	1153	218	1144.44	1153	50
		40	1142.17	1145	64	1142.17	1145	15
		45	1142	1142	0	1142	1142	0

to the asymmetric instance ftv170. This is clearly partly due to lack of difference in the linear programming relaxation value of the PQR formulation with the former constraints and of the PQR formulation with the latter constraints. Nevertheless, the branch-and-cut algorithm B&C$_1$ solved all nine instances in an average of 162 seconds, whereas the branch-and-cut algorithm B&C$_2$ in an average of 63 seconds, which corresponds to a reduction of more than 50%.

7.6 Computational experiment

In this section we present computational results to assess the performance of the branch-and-cut algorithm in solving the test instances described in Section 7.3 to optimality. We divide this section in two parts, the first corresponding to the results on asymmetric instances in Section 7.6.1 and the second corresponding to the results on symmetric instances in Section 7.6.2.

7.6.1 Results for asymmetric instances

This section presents the results obtained on the asymmetric test instances by using the branch-and-cut algorithm proposed in Section 7.4. Given the large number of test instances, we only present results for a subset of the instances of set B in the main body of text, and present the remaining results in Appendix A.1. For each instance, we present results for values of p which are multiples of 5, starting with $p = 5$ and up to the maximum possible value of p for that instance (e.g., for an instance with 40 nodes, the value of p can go up to 20). Additionally, we impose a time limit of 10800 seconds (three hours).

Tables 7.4 and 7.5 show the results with respect to obtaining the optimal solution, with Table 7.4 focusing on the instances ft70, ftv70 and kro120p, and Table 7.5 on the instance ftv170. Both tables have the following format. The first three columns indicate the name of the instance, the

Table 7.4: Optimal solution results for asymmetric instances (1 of 2)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
ft70	10	5	38120	3	98	851	62
		10	37978	0	0	0	0
		15	38033	1	0	0	114
		20	38390	7	99	5	976
		25	39233	9	79	0	1604
		30	40539	11	125	0	1460
		35	42908	2	0	0	166
ftv70	11	5	1826	2	47	474	7
		10	1766	0	0	0	0
		15	1771	0	0	0	56
		20	1841	1	0	0	86
		25	1978	1	1	0	243
		30	2210	2	5	0	469
		35	2535	1	0	0	156
kro124p	32	5	35435	13	218	1035	50
		10	35010	15	160	1848	71
		15	34799	68	1171	4924	207
		20	34433	62	397	5651	161
		25	34267	87	591	6107	358
		30	34002	1	0	52	6
		35	34050	0	0	0	0
		40	34310	0	0	0	9
		45	35331	1	0	0	39
		50	37541	1	0	0	53

number p' of circuits in the assignment relaxation and the value of p, respectively. The other five columns show, respectively, the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit was reached (OPT), the time taken in seconds (t), the number of branch-and-bound nodes explored (B&B), the number of violated $(\leq p)$ constraints found ($\#(\leq p)$) and the number of violated $(\geq p)$ constraints found ($\#(\geq p)$).

Tables 7.6 and 7.7 show the results with respect to obtaining the linear programming relaxation value, with Table 7.6 focusing on the instances ft70, ftv70 and kro120p, and Table 7.7 on the instance ftv170. Both tables have the following format. The first three columns indicate the name of the instance, the number p' of circuits in the assignment relaxation and the value of p, respectively. The fourth column indicates the optimal value of the instance (OPT) taken from either Table 7.4 or Table 7.5. The following five columns show, respectively, the linear program-

Table 7.5: Optimal solution results for asymmetric instances (2 of 2)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
ftv170	17	5	2683	14	23	339	7
		10	2636	8	0	148	0
		15	2631	0	0	0	0
		20	2631	0	0	0	0
		25	2639	28	5	1	455
		30	2658	44	7	38	825
		35	2704	290	206	1240	4304
		40	2736	508	24	0	3548
		45	2799	356	160	64	3758
		50	2884	735	53	0	4825
		55	3008	2470	135	10	8649
		60	3205	3648	267	0	9703
		65	[3428, 3432]	10800*	2491	31	17838
		70	[3684, 3706]	10800*	1253	0	14429
		75	[3767, 4091]	10800*	0	0	16046
		80	[4099, 4403]	10800*	0	0	15483
		85	4777	7647	3412	2	12350

*Not solved to optimality within the time limit of three hours

ming relaxation value (LP), the percentage of gap between the linear programming relaxation value and the best known upper bound (gap), the time taken to obtain the linear programming relaxation value (t_L) in seconds, the number of violated $(\leq p)$ constraints found ($\#(\leq p)$) and the number of violated $(\geq p)$ constraints found ($\#(\geq p)$).

We start by analyzing the results in Table 7.4. Notice that the branch-and-cut algorithm was able to solve the instances ft70, ftv70 and kro124p, for all values of p, in a maximum of 87 seconds. This suggests that for asymmetric instances with up to 100 nodes the branch-and-cut algorithm proposed is very effective. Recall the observation made regarding the distinction between the cases $p < p'$ and $p > p'$ in Section 7.4.1. In particular, observe that for $p < p'$ few violated $(\geq p)$ constraints are found, and the ones which are we believe can be mostly attributed to symmetry-breaking purposes. Conversely, for $p > p'$, very few violated $(\leq p)$ constraints are found. These results suggest that, in practice, we can expect different behavior in the branch-and-cut algorithm depending on whether $p < p'$ or $p > p'$. More precisely, the set of $(\leq p)$ constraints used have a much bigger influence in the former case, whereas the $(\geq p)$ constraints used have a much bigger influence in the latter.

The results of Table 7.5 confirm this last observation, apart from the case $p = 35$ in which there were a considerable number of violated $(\leq p)$ constraints found. For the instance ftv170,

Table 7.6: Linear programming relaxation results for asymmetric instances (1 of 2)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
ft70	10	5	38120	38055.6	0.17	0	89	1
		10	37978	37978	0.00	0	7	0
		15	38033	38018.5	0.04	0	0	59
		20	38390	38275.2	0.30	0	0	185
		25	39233	39027.9	0.52	1	0	417
		30	40539	40258.3	0.69	2	0	800
		35	42908	42297	1.42	0	0	119
ftv70	11	5	1826	1804.78	1.16	1	137	11
		10	1766	1766	0.00	0	8	10
		15	1771	1769.5	0.08	0	0	61
		20	1841	1837	0.22	0	0	130
		25	1978	1954.41	1.19	0	0	241
		30	2210	2140.94	3.12	0	0	285
		35	2535	2496.63	1.51	0	0	383
kro124p	32	5	35435	35114.9	0.90	0	229	3
		10	35010	34681.5	0.94	1	265	2
		15	34799	34421.8	1.08	1	215	1
		20	34433	34227.8	0.60	4	463	4
		25	34267	34083.1	0.54	0	69	7
		30	34002	33990	0.04	0	51	1
		35	34050	34050	0.00	0	0	13
		40	34310	34294.1	0.05	0	3	24
		45	35331	35082	0.70	0	0	28
		50	37541	36663	2.34	0	0	23

which has 171 nodes, the branch-and-cut algorithm proposed was able to solve most of the values of p, except for $p \in \{65, 70, 75, 80\}$. It is clear from these results that the value of p has a strong influence in the solution times. In particular, the solution times are lower in the cases in which p is close to the value of the assignment relaxation p', but are significantly higher as the difference between p and p' increases.

With respect to the results of Table 7.6, we can see that the linear programming relaxation values obtained are also directly influenced by the difference between p and p'. For the three instances ft70, ftv70 and kro124p, and excluding the cases in which the solution obtained was the same as the solution of the assignment relaxation, we can see a minimum and maximum gap of 0.04% and 3.12%, respectively, which is a considerable difference. As for the results of Table 7.7 with respect to the instance ftv170, we can draw similar conclusions. More precisely,

Table 7.7: Linear programming relaxation results for asymmetric instances (2 of 2)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
ftv170	17	5	2683	2662	0.78	1	50	0
		10	2636	2635	0.04	2	124	1
		15	2631	2631	0.00	1	27	7
		20	2631	2631	0.00	2	59	107
		25	2639	2635.14	0.15	16	2	440
		30	2658	2644.38	0.51	9	0	454
		35	2704	2671.92	1.19	56	0	1152
		40	2736	2714.81	0.77	103	0	1698
		45	2799	2772.13	0.96	603	1	3385
		50	2884	2844.78	1.36	1153	1	4950
		55	3008	2954.38	1.78	2083	0	7090
		60	3205	3105.34	3.11	4248	0	10647
		65	[3428, 3432]	3325.09	3.12	9716	0	14291
		70	[3684, 3706]	3567.63	3.73	10800*	0	16163
		75	[3767, 4091]	3727.48	8.89	10800*	0	15918
		80	[4099, 4403]	3962.44	10.01	10800*	0	15316
		85	4777	4644.5	2.77	1081	0	5890

*Not solved to optimality within the time limit of three hours

for values of p closer to p' the gap values are small, with a maximum of 1.19% for p up to 35, for instance. However, as the value of p increases, we observe considerably larger gap values such as, for example, in the cases $p = 60$ and $p = 85$, with gap values of 3.11% and 2.77%, respectively. Observe that for the unsolved cases the gap value reported is an upper bound on the real gap value if it were to be calculated with the (unknown) optimal solution. We believe that the values 8.89% and 10.01% for the cases $p = 75$ and $p = 80$, respectively, largely overestimate the real gap value since, as we can see in Table 7.5, no branch-and-bounds node were explored (i.e., the time limit was reached in the root node) and, therefore, the upper bounds are most likely far from the optimal value. Finally, we observe that the number of violated inequalities found is also consistent with the results analyzed before, that is, it depends on whether $p < p'$ or $p > p'$.

7.6.2 Results for symmetric instances

We now present the results obtained with respect to the symmetric test instances. Once again, given the large number of instances, we present some of the results in Appendix A.2.

Tables 7.8 and 7.9 show the results with respect to obtaining the optimal solution, with Table 7.8 focusing on the instances pr76 and rat99, and Table 7.9 on the instances kroB100 and

Table 7.8: Optimal solution results for symmetric instances (1 of 2)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
pr76	36	5	[96104, 97764]	10800*	130191	79582	738
		10	[90747, 91883]	10800*	61096	120131	944
		15	86380	1066	18155	46964	178
		20	82311	99	588	9705	35
		25	82040	1757	6100	39375	137
		30	[80612, 81961]	10800*	21436	54850	491
		35	77973	1	0	159	4
rat99	45	5	1237	8	222	1580	3
		10	1212	68	419	4726	46
		15	1195	39	391	5975	35
		20	1184	172	2674	25897	145
		25	1170	177	1177	18906	198
		30	1159	41	157	6544	47
		35	1153	50	419	9912	46
		40	1145	15	308	8129	35
		45	1142	0	0	0	0

*Not solved to optimality within the limit of three hours.

kroC100. Both tables have the following format. The first three columns indicate the name of the instance, the number p' of circuits in the assignment relaxation and the value of p, respectively. The other five columns show, respectively, the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit was reached (OPT), the time taken in seconds (t), the number of branch-and-bound nodes explored (B&B), the number of violated $(\leq p)$ constraints found ($\#(\leq p)$) and the number of violated $(\geq p)$ constraints found ($\#(\geq p)$).

Tables 7.10 and 7.11 show the results with respect to obtaining the linear programming relaxation value, with Table 7.10 focusing on the instances pr76 and rat99, and Table 7.11 on the instances kroB100 and kroC100. Both tables have the following format. The first three columns indicate the name of the instance, the number p' of circuits in the assignment relaxation and the value of p, respectively. The fourth column indicates the optimal value of the instance (OPT) taken from either Table 7.8 or Table 7.9. The following five columns show, respectively, the linear programming relaxation value (LP), the percentage of gap between the linear programming relaxation value and the best known upper bound (gap), the time taken to obtain the linear programming relaxation value (t_L) in seconds, the number of violated $(\leq p)$ constraints found ($\#(\leq p)$) and the number of violated $(\geq p)$ constraints found ($\#(\geq p)$).

We start with the results of Tables 7.8 and 7.9. We can see that the branch-and-cut algo-

Table 7.9: Optimal solution results for symmetric instances (2 of 2)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
kroB100	43	5	[21023, 21093]	10800*	104060	55636	4706
		10	[19869, 20289]	10800*	39475	120403	4976
		15	[19198, 19364]	10800*	30045	123221	3219
		20	18727	8587	23456	96014	1077
		25	18132	8559	13631	102279	1484
		30	17606	4426	7274	76271	921
		35	17210	109	490	8919	86
		40	16921	80	316	8506	62
		45	16838	2	0	329	2
		50	18684	2	0	0	0
kroC100	45	5	19998	7676	71857	64310	1693
		10	19077	1769	18122	38756	1126
		15	18454	1950	20767	34709	907
		20	17958	208	1504	16261	323
		25	17668	930	1144	24979	499
		30	[17421, 17539]	10800*	13164	93350	1604
		35	17189	5820	13899	54226	610
		40	16926	74	579	8571	104
		45	16801	0	0	0	0
		50	17738	2	0	0	0

*Not solved to optimality within the limit of three hours.

rithm was able to solve almost all values of p for the four instances, with the exception of $p \in \{5, 10, 30\}$ in the case of instance pr76, $p \in \{5, 10, 15\}$ in the case of instance kroB100 and $p = 30$ for instance kroC100. These results indicate that the performance of the branch-and-cut algorithm depends on the value of p, however, there is no clear pattern for this dependency. Nevertheless, the branch-and-cut algorithm is effective for symmetric instances with up to 100 nodes. Observe that the instances which we decided to present in the main body of text are the instances where the performance was worse on average, except for instance rat99, that is, they are the worst case scenario for the symmetric instances which we tested. There is a similar behavior to the results on asymmetric instances with respect to the distinction between $p < p'$ and $p > p'$. More precisely, for $p < p'$ the number of violated $(\geq p)$ constraints found is much lower than the number of violated $(\leq p)$ constraints found.

When compared to the results for the asymmetric instances of the previous section, it is clear that the branch-and-cut algorithm is able to solve similarly sized asymmetric instances more efficiently. Additionally, for asymmetric instances, the value of p' is usually low when compared

Table 7.10: Linear programming relaxation results for symmetric instances (1 of 2)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
pr76	36	5	[96104, 97764]	91255.6	6.66	0	135	0
		10	[90747, 91883]	85634	6.80	2	534	0
		15	86380	81211.5	5.98	0	58	0
		20	82311	79365.2	3.58	0	50	0
		25	82040	78332	4.52	0	52	1
		30	[80612, 81961]	77731.8	5.16	0	52	1
		35	77973	77207.3	0.98	1	671	0
rat99	45	5	1237	1228.5	0.69	1	297	0
		10	1212	1196.5	1.28	1	198	0
		15	1195	1178.17	1.41	1	218	0
		20	1184	1164.17	1.67	2	250	1
		25	1170	1154.38	1.34	1	272	0
		30	1159	1148	0.95	3	325	1
		35	1153	1144.44	0.74	2	146	0
		40	1145	1142.17	0.25	2	254	0
		45	1142	1142	0.00	0	30	0

to the maximum possible value, whereas in the case of symmetric instances we observe the reversed situation. Given the distinction between the $p < p'$ and $p > p'$ cases, we can conclude that the $(\geq p)$ constraints are more important in the asymmetric case and the $(\leq p)$ constraints in the symmetric case.

We believe that the difference in performance between asymmetric and symmetric instances is due to two main reasons. Firstly, it is possible to prove a result similar to Proposition 3 for the restricted multi-cut inequalities $q(S', i)+r(S', S)+p(i, S) \geq y_i$ (6.36), namely stating that these constraints are expected to provide lower linear programming relaxation values for symmetric cost instances which also negatively influences the cutting plane phase of the branch-and-cut algorithm. Note, however, that since the $(\leq p)$ constraints are more important in the symmetric case, this result does not fully explain the worse performance for symmetric instances and, in fact, we believe that the $(\leq p)$ constraints of the PQR formulation are not as good, in practice, as the $(\geq p)$ constraints, which are more important in the asymmetric case. Secondly, as we showed in Section 7.5.2, the use of the restricted multi-cut inequalities (6.36) for symmetry-breaking purposes substantially improved the results for asymmetric instances, whereas this improvement was not as noticeable for symmetric instances. This is related to the previous observation since the increase in the linear programming relaxation value was significant in asymmetric instances but non-existent in symmetric ones. Additionally, the restricted multi-cut inequalities (6.36) are the $(\geq p)$ constraints of the PQR formulation, hence, they are more

Table 7.11: Linear programming relaxation results for symmetric instances (2 of 2)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
kroB100	43	5	[21023, 21093]	20282.3	3.84	1	220	0
		10	[19869, 20289]	19230.5	5.22	1	297	0
		15	[19198, 19364]	18438	4.78	2	286	1
		20	18727	17849.5	4.69	1	114	3
		25	18132	17357.5	4.27	1	119	2
		30	17606	17057.8	3.11	1	138	1
		35	17210	16890.3	1.86	1	117	5
		40	16921	16834	0.51	112	2527	0
		45	16838	16830	0.05	1	78	0
		50	18684	16830	9.92	0	0	0
kroC100	45	5	19998	19016	4.91	1	208	0
		10	19077	18250	4.34	1	143	0
		15	18454	17713.5	4.01	0	92	0
		20	17958	17344.8	3.41	1	93	1
		25	17668	17116.3	3.12	2	245	0
		30	[17421, 17539]	16959.5	3.30	1	115	2
		35	17189	16863.5	1.89	24	1499	0
		40	16926	16808.6	0.69	2	240	8
		45	16801	16801	0.00	2	177	1
		50	17738	16801	5.28	0	0	0

relevant for the asymmetric case.

With respect to the results of Tables 7.10 and 7.11, we can see that the gap values observed in symmetric instances are substantially worse than in asymmetric instances. For example, for instance pr76 the lowest gap value was 0.98% but the second lowest was 3.58%. Since, as we observed before, the $(\leq p)$ constraints are more important in the symmetric instances and, as the results show, the number of violated $(\geq p)$ constraints found is negligible, it is clear that the $(\leq p)$ constraints of the PQR formulation do not provide good linear programming relaxations values, specially compared to the ones provided by the $(\geq p)$ constraints in the asymmetric case.

7.7 Additional computational results

This section provides additional computational results with two distinct aims. In Section 7.7.1 we provide a numerical comparison of the x-v formulation presented in Section 6.4 to the PQR formulation. Then, in Section 7.7.2 we present computational results that allow a comparison of our branch-and-cut algorithm based on the PQR formulation to other solution methods in the

Algorithm 7.5

Require: A point (x^*, v^*) and the original graph G with the addition of a node s with an outgoing arc to every node of V.

1: **for all** $i \in V$ **do**
2: Set the capacities of the arcs $\{(s,k) : k \in V\}$ to v_k^{*i}, the capacities of the arcs $(j,k) \in A$ to x_{jk}^*, and determine the maximum flow w from s to i.
3: **if** $w < 1$ **then**
4: The corresponding minimum cut defines a violated inequality (6.48) in which S is the subset of nodes in the same shore as node i.
5: **end if**
6: **end for**

literature with respect to the variant of the Hamiltonian p-median problem in which two-node circuits are not allowed.

7.7.1 Numerical comparison between the x-v formulation and the PQR formulation

In this section we provide computational results to compare the x-v formulation described in Section 6.4 to the PQR formulation. More precisely, we compare the branch-and-cut algorithm based on the PQR formulation presented in Section 7.4 to a branch-and-cut algorithm based on the x-v formulation. We will start by detailing the branch-and-cut algorithm based on the x-v formulation very briefly.

In terms of separation algorithms, we use an adaptation of algorithm 4.1 presented in Section 4.4.1, which is an adaptation itself of an algorithm presented by Erdoğan et al. (2016), to separate the $(\geq p)$ constraints $v_S^i + x_{ij} \leq v_S^j + 1$ (6.49) of the x-v formulation. As for the $(\leq p)$ constraints of the x-v formulation, the separation algorithm which we use is an adaptation of an algorithm described by Gollowitzer et al. (2014) and is similar to other algorithms presented in Section 4.4. For clarity, we present its details. Observe that, by using constraints $\sum_{j \in V} v_j^i = 1$ (6.46), the $(\leq p)$ constraints of the x-v formulation $x(S', S) \geq v_{S'}^i$ (6.48) can be written as follows:

$$x(S', S) + v_S^i \geq 1 \qquad \forall S \subset V,\ \forall i \in S. \tag{7.1}$$

This alternative form of writing constraints (6.48) indicates that we can separate them in an exact way by resorting to max-flow/min-cut computations in the original graph with the addition of a node s which has an outgoing arc to every node of V, as shown in algorithm 7.5.

We also use heuristic separation algorithms for both the $(\leq p)$ and $(\geq p)$ constraints based on determining connected components similarly to algorithms 7.2 and 7.4, respectively. The rest of the branch-and-cut algorithm based on the x-v formulation functions exactly the same as

the branch-and-cut algorithm based on the PQR formulation, including the use of the symmetry-breaking constraints (6.47) and other techniques described in Section 7.4.

For the comparison results we selected a subset of the test instances, both symmetric and asymmetric, to perform the comparison in the main body of text. The remaining results can be seen in Appendix A.3. We do not present results for the instance ftv170 since the branch-and-cut algorithm based on the x-v formulation was unable to solve it for the values of p tested.

Tables 7.12 and 7.13 show the comparison results, with Table 7.12 focusing on the asymmetric instances ftv70 and kro124p, and Table 7.13 on the symmetric instances kroB100 and kroC100. Both tables have the following format. The first three columns indicate the name of the instances, the number p' of circuits in the assignment relaxation and the value of p, respectively. The remaining six columns are divided into two parts, each with three columns corresponding to, respectively, the linear programming relaxation value (LP), the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit of 10800 seconds was reached (OPT) and the respective time taken (t) in seconds. The first part corresponds to the branch-and-cut algorithm described in Section 7.4 based on the PQR formulation, denoted by B&C PQR, and the second part corresponds to the branch-and-cut algorithm based on the x-v formulation described above and denoted by B&C x-v.

We will separate our evaluation on the reported results between asymmetric and symmetric instances. In the former case, which corresponds to the results of Table 7.12, and in terms of obtaining the optimal solution, the results show that the branch-and-cut algorithm B&C PQR outperforms the branch-and-cut algorithm B&C x-v. In fact, the branch-and-cut algorithm B&C PQR is able to obtain the optimal solution for all of the values of p for the two instances ftv70 and kro124p, whereas the branch-and-cut algorithm B&C x-v is unable to do so for six of them. In addition, regarding the values of p in which both branch-and-cut algorithms are able to obtain the optimal solution, the solution times of the branch-and-cut algorithm B&C PQR are substantially lower, with exception to a few cases. If we analyze the linear programming relaxation values, the PQR formulation provides higher linear programming relaxation values for the case where $p > p'$ and the x-v formulation for the case where $p < p'$. However, the difference in the linear programming relaxation values between the two models is greater in the $p > p'$ case.

Regarding the results for the symmetric instances reported in Table 7.13, and in contrast to the asymmetric case, the dominance of the performance of the branch-and-cut algorithm B&C PQR over the performance of the branch-and-cut algorithm B&C x-v is not as impressive. In terms of obtaining the optimal solution, the branch-and-cut algorithm B&C PQR was able to solve 16 out of 20 values of p for the instances kroB100 and kroC100, whereas the branch-and-cut algorithm B&C x-v only 12. However, the branch-and-cut algorithm B&C x-v was able to solve three values of p that the branch-and-cut algorithm B&C PQR was not, namely $p \in \{5, 10, 15\}$ for instance kroB100. In addition, in some cases, the solution times of the branch-

Table 7.12: Numerical comparison between the x-v and the PQR formulations (1 of 2)

			B&C PQR			B&C x-v		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
ftv70	11	5	1804.78	1826	2	1806.88	1826	3
		10	1766	1766	0	1766	1766	0
		15	1769.5	1771	0	1766.67	1771	25
		20	1837	1841	1	1783.44	[1790, 1990]	10800*
		25	1954.41	1978	1	1830.07	[1841, 2092]	10800*
		30	2140.94	2210	2	1896.28	[1766, 2320]	10800*
		35	2496.63	2535	1	1988.87	[2009, 2878]	10800*
kro124p	32	5	35114.9	35435	13	35212.9	35435	20
		10	34681.5	35010	15	34812.5	35010	41
		15	34421.8	34799	68	34523.6	34800	3320
		20	34227.8	34433	62	34280.5	34433	35
		25	34083.1	34267	87	34115.6	34267	43
		30	33990	34002	1	34002	34002	0
		35	34050	34050	0	34019.8	34050	3257
		40	34294.1	34310	0	34171.5	34310	8
		45	35082	35331	1	34487.2	[33991, 36206]	10800*
		50	36663	37541	1	34982.3	[33978, 46790]	10800*

*Not solved to optimality within the time limit of three hours

and-cut algorithm B&C x-v were lower than the solution times of the branch-and-cut algorithm B&C PQR. In general, and on average, the branch-and-cut algorithm B&C PQR performs better, however, the branch-and-cut algorithm B&C x-v is more competitive than in the asymmetric case. In terms of linear programming relaxation values, we have similar conclusions to the ones for the asymmetric case. In particular, for the case where $p > p'$ the PQR formulation provides higher linear programming relaxation values, whereas for the case such that $p < p'$ the x-v formulation provides higher linear programming relaxation values. The main difference in these results, and in contrast to the results of the asymmetric case, is that instances where $p < p'$ arise much more often in the symmetric case. This is the main reason for the increased competitiveness of the branch-and-cut algorithm B&C x-v since, as already noted before for the asymmetric case, the cases with $p > p'$ are the cases in which the PQR formulation provides higher linear programming relaxation values. Observe that we decided to choose the instances in which the branch-and-cut algorithm B&C x-v provides the best results to show in Table 7.5 and, in fact, the branch-and-cut algorithm B&C PQR has a substantially better average performance when considering the complete set of symmetric test instances.

Summarizing, in general the branch-and-cut algorithm based on the PQR formulation per-

Table 7.13: Numerical comparison between the x-v and the PQR formulations (2 of 2)

			B&C PQR			B&C x-v		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
kroB100	43	5	20282.3	[21034, 21093]	10800*	20495.5	21082	1995
		10	19230.5	[19872, 20289]	10800*	19635.9	20127	3929
		15	18438	[19204, 19364]	10800*	18944.9	19307	1775
		20	17849.5	18727	8587	18324.2	18727	305
		25	17357.5	18132	8559	17809.9	18132	93
		30	17057.8	17606	4426	17395.5	17606	21
		35	16890.3	17210	109	17048.2	17210	30
		40	16834	16921	80	16861.8	16921	8
		45	16830	16838	2	16830	16838	3
		50	16830	18684	2	16830	[16830, 21852]	10800*
kroC100	45	5	19016	19998	7676	19357.2	[19421, 21265]	10800*
		10	18250	19077	1769	18618.2	[18760, 19211]	10800*
		15	17713.5	18454	1950	18058	[18300, 18651]	10800*
		20	17344.8	17958	208	17670	[17878, 17958]	10800*
		25	17116.3	17668	930	17367.8	[17528, 17668]	10800*
		30	16959.5	[17434, 17539]	10800*	17118.9	[17379, 17476]	10800*
		35	16863.5	17189	5820	16940.5	17189	123
		40	16808.6	16926	74	16816.5	16926	23
		45	16801	16801	0	16801	16801	0
		50	16801	17738	2	16801	[16801, 21066]	10800*

*Not solved to optimality within the time limit of three hours

forms better than the branch-and-cut algorithm based on the x-v formulation. This dominance is more noticeable for the asymmetric instances, however, even in the symmetric instances, the branch-and-cut algorithm based on the PQR formulation is able to solve more instances than the branch-and-cut algorithm based on the x-v model, specially if you consider the complete set of instances. We believe that the advantages of the PQR formulation are the efficiency of the separation of the restricted multi-cut inequalities $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.36) and the good linear programming relaxation values for $p > p'$. For the symmetric instances the case $p > p'$ is rare and, thus, the latter advantage is lost. Nevertheless, the efficiency of the separation of the restricted multi-cut inequalities (6.36) allows the branch-and-cut algorithm based on the PQR formulation to outperform, in general, the branch-and-cut algorithm based on the x-v formulation.

7.7.2 Results for the variant in which two-node circuits are not allowed

As we explained in the introduction of Chapter 5, most of the recent literature on the Hamiltonian p-median problem, in particular the works by Gollowitzer et al. (2014), Erdoğan et al. (2016) and Marzouk et al. (2016), studies a variant of the Hamiltonian p-median problem in which two-node circuits are not allowed and only for symmetric cost instances. In this section we show how to adapt the PQR formulation to handle this case and present computational results to compare our approach to the previous ones in the literature.

Possibly the most trivial way to prevent two-node circuits in the PQR formulation is to add the following constraints:

$$p_{ij} + p_{ji} + q_{ij} + q_{ji} + r_{ij} + r_{ji} \leq 1 \qquad \forall i, j \in V,\ i \neq j. \tag{7.2}$$

The interpretation of the p, q and r variables also allows for other ways to model this situation as, for instance, shown by the following set of constraints:

$$\sum_{k \neq j} q_{ki} \geq p_{ij} \qquad \forall i, j \in V,\ i \neq j. \tag{7.3}$$

Note that constraints (7.2) state that an arc $(i, j) \in A$ cannot be used in both directions simultaneously, whereas constraints (7.3) ensure that if an arc $(i, j) \in A$ is used in which i is an acting depot, then the ingoing arc to i must come from a node k such that $k \neq j$. It can be easily shown that, with respect to the linear programming relaxation, one set does not dominate the other, however, their behavior is similar in the sense that earlier testing showed that both lead to poor performances of the branch-and-cut algorithm based on the PQR formulation.

Recall, however, that an idea similar to the one of constraints (7.3) was used in the symmetry-breaking constraints (6.38) presented in Section 6.3.2 used to prevent alternative solutions due to reversed circuits having the same cost in symmetric instances. In particular, the symmetry-breaking constraints (6.38) state that if a p-arc $(i, j) \in A$ is used, then the ingoing q-arc to i must come from a node $k \in V$ such that $k \geq j$. In the Hamiltonian p-median problem we had to explicitly consider the case $k = j$, otherwise we could eliminate circuits with only two nodes, however, this is not the case in the variant of the problem in which two-node circuits are not allowed. Therefore, consider the following constraints:

$$\sum_{k > j} q_{ki} \geq \sum_{k \geq j} p_{ik}, \qquad \forall i, j \in V,\ i \neq j. \tag{7.4}$$

The difference when compared to the symmetry-breaking constraints (6.38) is that on the left-hand side we remove the case $k = j$ since we do not want circuits with only two nodes. Thus, constraints (7.4) not only implicitly prevent the existence of two-node circuits, but are also symmetry-breaking constraints for reversed alternative circuits.

Table 7.14: Evaluating the effectiveness of the non-trivial two-node circuit elimination constraints

			B&C$_1$			B&C$_2$		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
rat99	8	9	1203.54	1209.09	76	1203.98	1209.09	12
		14	1203.54	1224.1	3079	1214.61	1224.1	23
		19	1203.99	1245.16	7102	1235.13	1245.16	43
		24	1209.71	[1267.22, 1273.23]	10800*	1263.17	1273.23	44
		33	1238.64	[1309.4, 1397.26]	10800*	1343.4	1373.37	3003

*Not solved to optimality within the time limit of three hours

We also showed that, in the presence of symmetry-breaking constraints of type I, for instance the restricted multi-cut inequalities $q(S', i) + r(S', S) + p(i, S) \geq y_i$ (6.36), the symmetry-breaking constraints (6.38), and consequently constraints (7.4), are only relevant for circuits with more than two nodes. Since in the Hamiltonian p-median problem, in which two-node circuits are allowed, the solutions observed are comprised of many circuits with only two nodes, the symmetry-breaking constraints (6.38) only provided slight improvements in the overall branch-and-cut algorithm. However, and as we will see in Table 7.14, constraints (7.4) are extremely effective for this case since, by the definition of this variant of the Hamiltonian p-median problem, every circuit will be comprised of at least three nodes.

Table 7.14 presents computational results that show the effectiveness of constraints (7.4) for the symmetric instance rat99 with the following format. The first three columns indicate the name of the instance, the number p' of circuits in the assignment relaxation and the value of p used, respectively. The remaining six columns are divided into two parts, each with three columns corresponding to, respectively, the linear programming relaxation value (LP), the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit of 10800 seconds was reached (OPT) and the respective time taken (t) in seconds. The first part corresponds to the branch-and-cut algorithm based on the PQR formulation, denoted by B&C$_1$, which uses the trivial constraints (7.2) but not the symmetry-breaking constraints (6.38), and the second part corresponds to the branch-and-cut algorithm based on the PQR formulation, denoted by B&C$_2$, which uses constraints (7.4). Observe that constraints (7.3) could have been used instead of constraints (7.2), however, similar results would be observed for the branch-and-cut algorithm B&C$_1$.

The results of Table 7.14 show that the use of the symmetry-breaking/two-node circuit elimination constraints (7.4) is very effective in the variant of the Hamiltonian p-median problem in which two-node circuits are not allowed. In particular, the values of $p = 24$ and $p = 33$ went from unsolved by the branch-and-cut algorithm B&C$_1$ to solved by the branch-and-cut algorithm B&C$_2$. Additionally, reductions in the solution times where observed for the instances

Table 7.15: Optimal solution results for symmetric instances (two-node circuits not allowed) (1 of 2)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
eil76	4	7	542.954	10	17	168	338
		10	545.021	38	217	681	1928
		15	552.149	58	322	482	2859
		19	563.955	133	545	658	4978
		25	[590.395, 612.852]	10800*	6257	1646	39658
gr96	14	5	153568	119	1103	4565	553
		20	151403	77	23	1624	825
rat99	8	9	1209.09	12	16	187	104
		14	1224.1	23	79	408	352
		19	1245.16	43	22	382	868
		24	1273.23	44	64	354	995
		33	1373.37	3003	6980	2866	14681
kroA100	13	10	19900.9	151	1405	6723	385
		14	19637.5	95	124	4110	428
		20	19868.6	30	55	681	440
		25	20279.5	51	84	1275	583
		33	[21498, 23591]	10800*	4142	3130	31147
kroB100	20	10	20823.1	145	1014	7759	860
		14	20762.9	143	454	5317	607
		20	20660	75	26	2726	185
		25	20786.9	16	3	87	274
		33	[22204.6, 24968.4]	10800*	3915	2587	33454

*Not solved to optimality within the limit of three hours.

which had been solved by the branch-and-cut algorithm B&C$_1$. Without this set of constraints, the branch-and-cut algorithm based on the PQR formulation would simply not be competitive with the other methods from the literature.

We now present a more comprehensive set of results. For these results we used a subset of the instance set that Erdoğan et al. (2016) use in their work, which is comprised of the TSPLIB symmetric instances of set A described in Section 7.3 with the addition of instance u159 that has 159 nodes. Additionally, we use the same values of p used by Erdoğan et al. (2016). The results were obtained with the branch-and-cut algorithm based on the PQR formulation described in Section 7.4 with constraints (7.4) replacing the symmetry-breaking constraints (6.38). Given the large number of test instances, we present some of the results in Appendix A.4.

Tables 7.15 and 7.16 show the results with respect to obtaining the optimal solution, with Table 7.8 focusing on the instances eil76, gr96, rat99, kroA100 and kroB100, and Table 7.9

Table 7.16: Optimal solution results for symmetric instances (two-node circuits not allowed) (2 of 2)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
kroC100	13	10	19923.3	98	721	5063	526
		14	19938.8	67	231	4141	1006
		20	20135	55	223	1120	890
		25	20428	436	3542	3698	1959
		33	[21536.6, 23759]	10800*	4767	5350	25233
kroD100	14	10	20270.6	48	108	2775	212
		14	20267.2	42	22	1347	147
		20	20457	197	770	4308	2070
		25	20671.2	160	335	2803	2090
		33	[21720.3, 22439.7]	10800*	8916	4162	25151
kroE100	12	10	20766.4	25	48	2164	297
		14	20777.7	37	42	1529	449
		20	20937.4	65	102	1677	1251
		25	21174.9	92	130	2372	1114
		33	[22470.1, 22843.6]	10800*	9225	5026	20484
rd100	14	10	7524.08	147	751	5527	662
		14	7500.44	57	60	2449	105
		20	7537.98	101	411	4456	1131
		25	7555.83	42	11	721	699
		33	[7919.09, 8211.2]	10800*	6539	3624	27475
u159	20	5	41695	1194	3978	8017	312
		30	41723	540	187	8432	306

*Not solved to optimality within the limit of three hours.

on the instances kroC100, kroD100, kroE100, rd100 and u159. Both tables have the following format. The first three columns indicate the name of the instance, the number p' of circuits in the assignment relaxation and the value of p, respectively. The other five columns show, respectively, the optimal value obtained or the final interval of best lower and upper bounds obtained if the time limit of 10800 seconds was reached (OPT), the time taken in seconds (t), the number of branch-and-bound nodes explored (B&B), the number of violated $(\leq p)$ constraints found ($\#(\leq p)$) and the number of violated $(\geq p)$ constraints found ($\#(\geq p)$).

Tables 7.17 and 7.18 show the results with respect to obtaining the linear programming relaxation value, with Table 7.17 focusing on the instances eil76, gr96, rat99, kroA100 and kroB100, and Table 7.18 on the instances kroC100, kroD100, kroE100, rd100 and u159. Both tables have the following format. The first three columns indicate the name of the instance, the number p' of circuits in the assignment relaxation and the value of p, respectively. The

Table 7.17: Linear programming relaxation results for symmetric instances (two-node circuits not allowed) (1 of 2)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
eil76	4	7	542.954	541.493	0.27	4	224	415
		10	545.021	543.297	0.32	5	110	660
		15	552.149	547.805	0.79	12	224	1001
		19	563.955	553.543	1.85	14	257	1172
		25	[590.395, 612.852]	572.847	6.53	30	323	1674
gr96	14	5	153568	151513	1.34	1	312	0
		20	151403	150339	0.70	18	684	494
rat99	8	9	1209.09	1203.98	0.42	2	199	100
		14	1224.1	1214.61	0.78	5	329	231
		19	1245.16	1235.13	0.81	23	335	725
		24	1273.23	1263.17	0.79	36	406	1179
		33	1373.37	1343.4	2.18	32	286	1483
kroA100	13	10	19900.9	19570.2	1.66	11	680	3
		14	19637.5	19380.7	1.31	6	635	106
		20	19868.6	19523.3	1.74	20	974	290
		25	20279.5	19815.7	2.29	21	920	356
		33	[21498, 23591]	20542.7	12.92	106	1726	627
kroB100	20	10	20823.1	20444.8	1.82	3	507	4
		14	20762.9	20396.3	1.77	7	679	2
		20	20660	20414	1.19	18	1151	43
		25	20786.9	20581.7	0.99	49	1588	448
		33	[22204.6, 24968.4]	21413.9	14.24	39	1306	336

*Not solved to optimality within the limit of three hours.

fourth column indicates the optimal value of the instance (OPT) taken from either Table 7.15 or Table 7.16. The following five columns show, respectively, the linear programming relaxation value (LP), the percentage of gap between the linear programming relaxation value and the best known upper bound (gap), the time taken to obtain the linear programming relaxation value (t_L) in seconds, the number of violated ($\leq p$) constraints found ($\#(\leq p)$) and the number of violated ($\geq p$) constraints found ($\#(\geq p)$).

A comparison of our results to those reported by Erdoğan et al. (2016) shows that we have been able to optimally solve instances that Erdoğan et al. (2016) have not, namely the instance pr76 for $p = 15$ (reported in Appendix A.4), the instance gr96 for $p = 20$, the instance rat99 for $p \in \{14, 19, 24, 33\}$ and the instance u159 for $p \in \{5, 30\}$. Conversely, there are other instances which we have been unable to solve to optimality that Erdoğan et al. (2016) have, namely the

Table 7.18: Linear programming relaxation results for symmetric instances (two-node circuits not allowed) (2 of 2)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
kroC100	13	10	19923.3	19703.3	1.10	5	695	2
		14	19938.8	19725.6	1.07	14	725	296
		20	20135	19853.1	1.40	39	1065	511
		25	20428	20033.8	1.93	28	813	692
		33	[21536.6, 23759]	20286	14.62	30	899	672
kroD100	14	10	20270.6	19957	1.55	3	462	6
		14	20267.2	19962.9	1.50	12	825	57
		20	20457	20021.2	2.13	46	1086	495
		25	20671.2	20156.9	2.49	52	971	455
		33	[21720.3, 22439.7]	20669.6	7.89	64	1802	487
kroE100	12	10	20766.4	20651	0.56	3	451	1
		14	20777.7	20641	0.66	5	467	213
		20	20937.4	20715	1.06	18	859	444
		25	21174.9	20891.6	1.34	37	920	632
		33	[22470.1, 22843.6]	21485.3	5.95	40	1271	450
rd100	14	10	7524.08	7338.77	2.46	4	648	3
		14	7500.44	7336.96	2.18	20	1142	28
		20	7537.98	7354.05	2.44	33	1424	216
		25	7555.83	7419.3	1.81	37	1103	511
		33	[7919.09, 8211.2]	7670.45	6.59	35	1184	443
u159	20	5	41695	41079	1.48	3	236	1
		30	41723	41071.4	1.56	309	2667	81

*Not solved to optimality within the limit of three hours.

instance att48 for $p = 16$ (reported in Appendix A.4), instance eil76 with $p = 25$ and instance kroE100 with $p = 33$.

Regarding a comparison to the work of Marzouk et al. (2016), we observe that the results reported by Marzouk et al. (2016) show that their branch-and-price algorithm is not able to easily solve instances with small values of p, whereas the branch-and-cut algorithm based on the PQR formulation we propose is. In particular, we have been able to optimally solve a number of instances that were not solved by Marzouk et al. (2016) such as the instance pr76 for $p = 15$, the instance gr96 for $p = 5$ and the instances kroA100, kroB100 and kroD100 for $p \in \{10, 14\}$. In contrast, some instances optimally solved by Marzouk et al. (2016) proved difficult to solve with our branch-and-cut algorithm such as the instance eil76 for $p = 25$. In addition, Marzouk et al. (2016) tested many more values of p which we have not, thus, for those cases we do not

have a comparison.

The PQR formulation was created to solve the original Hamiltonian p-median problem as described by Branco & Coelho (1990), however, the results presented in this section suggest that the branch-and-cut algorithm based on the PQR formulation we proposed in this dissertation is able to compete with the current state-of-the-art methods for the variant of the problem in which two-node circuits are not allowed. In fact, our approach was able to provide optimal solutions for instances which had never been solved in the literature.

7.8 Concluding remarks

In this chapter we proposed a branch-and-cut algorithm based on the PQR formulation and used a large set of well-known traveling salesman problem instances, both symmetric and asymmetric, to test it. The performance of the branch-and-cut algorithm was satisfactory. In particular, the branch-and-cut algorithm was able to solve asymmetric instances with up to 100 nodes in less than two minutes and several values of p for an instance with 171 nodes within the time limit. As for symmetric instances, the branch-and-cut algorithm proved to be less effective than in the asymmetric case. Nevertheless, it was able to solve most values of p with respect to instances with up to 100 nodes.

We identified two advantages of the branch-and-cut algorithm based on the PQR formulation that explain its performance. Firstly, the separation of the $(\geq p)$ constraints is very efficient. Secondly, the linear programming relaxation values for the cases in which $p > p'$, which we recall are the cases in which the $(\geq p)$ constraints are more relevant, were relatively close to the optimal value. These advantages were more important in the asymmetric instances given that the case $p > p'$ arises less often in the symmetric instances, which is a possible explanation for the difference in performance for both cost structures. A disadvantage of the PQR formulation is that the linear programming relaxation values for $p < p'$, in which the $(\leq p)$ constraints are more relevant, were not as good as the ones for the $p > p'$ case, thus, a possible way to improve the results of the branch-and-cut algorithm is by finding additional $(\leq p)$ constraints which can improve the lower bounds for the $p < p'$ case.

As a complement to the discussion of Chapter 6, we also compared our approach to two other approaches. Firstly, we compared our branch-and-cut algorithm to a similar branch-and-cut algorithm based on the x-v formulation presented in Section 6.4.1. The results showed that the branch-and-cut algorithm based on the PQR formulation outperforms the branch-and-cut algorithm based on the x-v formulation in asymmetric instances. With respect to the symmetric instances, the branch-and-cut algorithm based on the PQR formulation was not as dominant, specially since the $(\leq p)$ constraints, which are more important in symmetric instances, of the x-v model were able to provide higher linear programming relaxation values than those of the PQR

formulation. Nevertheless, the branch-and-cut algorithm based on the PQR still outperforms the branch-and-cut algorithm based on the x-v formulation on average for symmetric instances.

Secondly, given that no state-of-the-art recent algorithms for the Hamiltonian p-median problem exist, we decided to adapt the PQR formulation and the respective branch-and-cut algorithm to the variant of the Hamiltonian p-median problem in which two-node circuits are not allowed, for which two recent state-of-the-art algorithms by Erdoğan et al. (2016) and Marzouk et al. (2016) exist. The results showed that our approach is competitive with both methods. More precisely, the adaptation of the branch-and-cut algorithm based on the PQR formulation to this variant was able to provide optimal solutions for previously unsolved instances. Even though the variant solved in this case is different from the Hamiltonian p-median problem as defined by Branco & Coelho (1990), these results indicate that our approach is certainly competitive with potential future algorithms which may be suggested for the Hamiltonian p-median problem.

Conclusion

In this dissertation we studied two optimization problems: the multi-depot routing problem and the Hamiltonian p-median problem, respectively in the first part and in the second part of the dissertation.

The objective of the multi-depot routing problem is, given a set of depots and a set of clients, to find a set of routes with minimum total cost, one for each depot, such that each client is visited in one and only one route and such that each route starts and ends at the same depot. The requirement that routes start and end at the same depot arises in most routing problems with multiple depots and it is modeled by so-called path elimination constraints. One of the contributions of this dissertation was the study of path elimination constraints in the multi-depot routing problem setting, which can be adapted to related problems.

We proposed a new set of exponentially-many multi-cut path elimination constraints which result from the projection of an arc-depot assignment variable based system of inequalities which had been previously used in the literature. The new multi-cut constraints can be efficiently separated in an exact way by resorting to max-flow/min-cut computations in an adequate 3-layered graph. Based on these constraints, we presented a branch-and-cut algorithm to solve the multi-depot routing problem. The results obtained by the branch-and-cut algorithm showed that the new multi-cut constraints are very effective and allowed to obtain optimal solutions of instances with up to 300 clients and 60 depots in a time limit of three hours. For symmetric cost instances the multi-cut constraints do not improve the linear programming relaxation values of the assignment relaxation of the problem, hence, their effectiveness was more evident in asymmetric instances. Nevertheless, due to their efficient separation, we were still able to obtain results on symmetric instances which do not differ significantly than the ones for asymmetric instances despite using formulations based on directed graphs.

We also presented many additional systems of inequalities to model path elimination constraints. Some of these systems of inequalities were based on similar variables used in the context of the precedence constrained (asymmetric) traveling salesman problem, namely client-depot assignment variables in this context, and some of the intuition was based on earlier studies in the literature based on this problem and on the Hamiltonian p-median problem. We also presented sets of path elimination constraints using the same client-depot assignment variables which, as

far as we know, are new and are based on the relationship of these variables to the arc-depot assignment variables used previously. Another important contribution of this dissertation was a new formulation presented which expanded the concept of the systems of inequalities based on the arc-depot assignment variables. In order to derive the new formulation we started by presenting a double network-flow system based on an earlier study for the traveling salesman problem and then we projected this system of inequalities on the space of the arc variables, the client-depot assignment variables and the arc-depot assignment variables. The earlier indication for this formulation is that, based on the computational results which showed that the linear programming relaxation values obtained were close to the optimal value in the instances tested, this formulation is an important formulation in order to, in the future, derive additional constraints in lower-dimensional spaces.

In the second part of this dissertation we studied the Hamiltonian p-median problem in which, given a set of nodes, we wish to find p circuits with minimum total cost such that each node is one and only one circuit. Based on the concept of acting depot which assigns nodes to be artificial depots and which had been used previously in the literature, we established a relationship to the multi-depot routing problem. This relationship allowed us to adapt, in a non-straightforward way, the new multi-cut path elimination constraints proposed for the multi-depot routing problem. This resulted in a new formulation based on a novel idea in which the information on which nodes are the acting depots is associated with the arc variables. Additionally, this new formulation was able to incorporate very effective symmetry-breaking constraints in its adaptation of the multi-cut constraints which greatly contributed to the computational results. With the new formulation we developed another branch-and-cut algorithm that was able to solve asymmetric instances with up to 171 nodes and symmetric instances with up to 100 nodes. Since in the recent literature most algorithmic work is based on a variant of the Hamiltonian p-median problem in which two-node circuits are not allowed, we also showed how to adapt our formulation and the corresponding branch-and-cut algorithm to solve this case. The results showed that we were able to solve instances which were previously unsolved in the literature.

Summarizing, the main contributions of the dissertation were the following: (i) we studied path elimination constraints which are an important set of constraints arising in the literature in routing problems involving multiple depots; (ii) we presented, in particular, a new set of multi-cut path elimination constraints which can be separated very efficiently; (iii) we presented a new formulation based on arc variables and on two sets of depot assignment variables which is able to provide very good linear programming relaxation values; (iv) we implicitly showed that the new multi-cut constraints are very versatile since they can be adapted to other problems, even if not completely straightforwardly, by (v) presenting a new formulation for the Hamiltonian p-median problem and a corresponding branch-and-cut algorithm which is able to solve instances of considerable dimension and which is competitive with state-of-the-art algorithms proposed

for the variant of the Hamiltonian p-median problem in which two-node circuits are not allowed.

This dissertation opens at least three directions for future investigations. Firstly, we introduced a study for the multi-depot routing problem which looked at combining constraints in higher-dimensional spaces in order to obtain constraints in lower-dimensional spaces, however, this introduction mainly looked at providing tools to perform these derivations. In particular, we believe that there is potential to find constraints in the space of the arc variables that combine subtour elimination constraints with path elimination constraints. Secondly, the formulation based on arc variables and on two sets of depot assignment variables provides linear programming relaxation values which were close to the optimal value in the instances tested. This means that studying the projection of the polyhedron defined by this system of inequalities can produce interesting constraints in lower-dimensional spaces, which also links to the previous future investigation proposed. Finally, the new formulation presented for the Hamiltonian p-median problem can be seen as an extension of formulations based on standard arc variables and acting depot variables. Therefore, another interesting investigation is to study the projection of the new formulation onto the space of the arc variables. For instance, one could start by studying its projection onto the space of both the arc variables and the acting depot variables in a intermediate step and then onto the space of the arc variables alone.

Bibliography

Albareda-Sambola, M. (2015), Location-routing and location-arc routing, *in* G. Laporte, S. Nickel & F. Saldanha da Gama, eds, 'Location Science', Springer, pp. 399–418.

Albareda-Sambola, M., Díaz, J. A. & Fernández, E. (2005), 'A compact model and tight bounds for a combined location-routing problem', *Computers & Operations Research* **32**(3), 407–428.

Applegate, D. L., Bixby, R. E., Chvátal, V. & Cook, W. J. (2006), *The traveling salesman problem: a computational study*, Princeton University Press.

Araque G., J. R., Kudva, G., Morin, T. L. & Pekny, J. F. (1994), 'A branch-and-cut algorithm for vehicle routing problems', *Annals of Operations Research* **50**(1), 37–59.

Balas, E., Fischetti, M. & Pulleyblank, W. R. (1995), 'The precedence-constrained asymmetric traveling salesman polytope', *Mathematical Programming* **68**(1), 241–265.

Bektaş, T. (2006), 'The multiple traveling salesman problem: an overview of formulations and solution procedures', *Omega* **34**(3), 209–219.

Bektaş, T. (2012), 'Formulations and Benders decomposition algorithms for multidepot salesmen problems with load balancing', *European Journal of Operational Research* **216**(1), 83–93.

Bektaş, T., Gouveia, L. & Santos, D. (2017), 'New path elimination constraints for multi-depot routing problems', *Networks* **70**(3), 246–261.

Bektaş, T., Gouveia, L. & Santos, D. (2019), 'Revisiting the Hamiltonian p-median problem: a new formulation on directed graphs and a branch-and-cut algorithm', *European Journal of Operational Research* **276**(1), 40–64.

Belenguer, J.-M., Benavent, E., Prins, C., Prodhon, C. & Wolfler Calvo, R. (2011), 'A branch-and-cut method for the capacitated location-routing problem', *Computers & Operations Research* **38**(6), 931–941.

Benavent, E. & Martínez-Sykora, A. (2013), 'Multi-depot multiple TSP: a polyhedral study and computational results', *Annals of Operations Research* **207**(1), 7–25.

Branco, I. M. & Coelho, J. D. (1990), 'The Hamiltonian p-median problem', *European Journal of Operational Research* **47**(1), 86–95.

Campêlo, M., Corrêa, R. & Frota, Y. (2004), 'Cliques, holes and the vertex coloring polytope', *Information Processing Letters* **89**(4), 159–164.

Dantzig, G., Fulkerson, D. & Johnson, S. (1954), 'Solution of a large-scale traveling-salesman problem', *Journal of the Operations Research Society of America* **2**(4), 393–410.

Desrochers, M. & Laporte, G. (1991), 'Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints', *Operations Research Letters* **10**(1), 27–36.

Erdoğan, G., Laporte, G. & Rodríguez-Chía, A. M. (2016), 'Exact and heuristic algorithms for the Hamiltonian p-median problem', *European Journal of Operational Research* **253**(2), 280–289.

Fernández, E. & Rodríguez-Pereira, J. (2016), 'Multi-depot rural postman problems', *TOP* pp. 1–33.

Fischetti, M., Salazar-González, J.-J. & Toth, P. (1997), 'A branch-and-cut algorithm for the symmetric generalized traveling salesman problem', *Operations Research* **45**(3), 378–394.

Fisher, M. L. & Jaikumar, R. (1981), 'A generalized assignment heuristic for vehicle routing', *Networks* **11**(2), 109–124.

Gavish, B. & Graves, S. C. (1978), 'The travelling salesman problem and related problems', *Working paper OR-078-78, Operations Research Center Working Papers, Massachusetts Institute of Technology, Operations Research Center* .

Glaab, H. & Pott, A. (2000), 'The Hamiltonian p-median problem', *The Electronic Journal of Combinatorics* **7**(1), 42.

Godinho, M. T., Gouveia, L. & Pesneau, P. (2011), On a time-dependent formulation and an updated classification of ATSP formulations, *in* A. R. Mahjoub, ed., 'Progress in Combinatorial Optimization', ISTE-Wiley, pp. 223—254.

Goldberg, A. V. & Tarjan, R. E. (1988), 'A new approach to the maximum-flow problem', *Journal of the ACM* **35**(4), 921–940.

Gollowitzer, S., Gouveia, L., Laporte, G., Pereira, D. L. & Wojciechowski, A. (2014), 'A comparison of several models for the Hamiltonian p-median problem', *Networks* **63**(4), 350–363.

Gouveia, L. & Pesneau, P. (2006), 'On extended formulations for the precedence constrained asymmetric traveling salesman problem', *Networks* **48**(2), 77–89.

Gouveia, L., Pesneau, P., Ruthmair, M. & Santos, D. (2018), 'Combining and projecting flow models for the (precedence constrained) asymmetric traveling salesman problem', *Networks* **71**(4), 451–465.

Gouveia, L. & Pires, J. M. (1999), 'The asymmetric travelling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints', *European Journal of Operational Research* **112**(1), 134–146.

Gouveia, L. & Pires, J. M. (2001), 'The asymmetric travelling salesman problem: on generalizations of disaggregated Miller-Tucker-Zemlin constraints', *Discrete Applied Mathematics* **112**(1), 129–145.

Hill, A. & Voß, S. (2016), 'Optimal capacitated ring trees', *EURO Journal on Computational Optimization* **4**(2), 137–166.

Hupp, L. & Liers, F. (2013), 'A polyhedral study of the Hamiltonian p-median problem', *Electronic Notes in Discrete Mathematics* **41**, 213–220.

IBM (2014), 'IBM ILOG CPLEX Optimization Studio 12.6.1', `https://www.ibm.com/analytics/cplex-optimizer`.

Langevin, A., Soumis, F. & Desrosiers, J. (1990), 'Classification of travelling salesman problem formulations', *Operations Research Letters* **9**(2), 127–132.

Laporte, G., Nobert, Y. & Arpin, D. (1984), 'Optimal solutions to capacitated multidepot vehicle routing problems', *Congressus Numerantium* **44**, 283–292.

Laporte, G., Nobert, Y. & Arpin, D. (1986), 'An exact algorithm for solving a capacitated location-routing problem', *Annals of Operations Research* **6**, 293–310.

Laporte, G., Nobert, Y. & Pelletier, P. (1983), 'Hamiltonian location problems', *European Journal of Operational Research* **12**(1), 82–89.

Laporte, G., Nobert, Y. & Taillefer, S. (1988), 'Solving a family of multi-depot vehicle routing and location-routing problems', *Transportation Science* **22**(3), 161–172.

Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R. & Shmoys, D. B. (1985), *The traveling salesman problem: a guided tour of combinatorial optimization*, John Wiley & Sons.

Marzouk, A. M., Moreno-Centeno, E. & Üster, H. (2016), 'A branch-and-price algorithm for solving the Hamiltonian p-median problem', *INFORMS Journal on Computing* **28**(4), 674–686.

Öncan, T., Altınel, İ. K. & Laporte, G. (2009), 'A comparative analysis of several asymmetric traveling salesman problem formulations', *Computers & Operations Research* **36**(3), 637–654.

Padberg, M. & Rinaldi, G. (1987), 'Optimization of a 532-city symmetric traveling salesman problem by branch and cut', *Operations Research Letters* **6**(1), 1–7.

Padberg, M. & Rinaldi, G. (1991), 'A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems', *SIAM Review* **33**(1), 60–100.

Roberti, R. & Toth, P. (2012), 'Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison', *EURO Journal on Transportation and Logistics* **1**(1), 113–133.

Sundar, K. & Rathinam, S. (2017), 'Multiple depot ring star problem: a polyhedral study and an exact algorithm', *Journal of Global Optimization* **67**(3), 527–551.

Toth, P. & Vigo, D. (2014), *Vehicle routing: problems, methods, and applications*, Society for Industrial and Applied Mathematics.

Wolsey, L. A. (1998), *Integer Programming*, John Wiley & Sons.

Wong, R. T. (1980), Integer programming formulations of the travelling salesman problem, *in* 'Proceedings of the IEEE International Conference of Circuits and Computers', IEEE Press Piscataway, NJ, pp. 149–152.

Zohrehbandian, M. (2007), 'A new formulation of the Hamiltonian p-median problem', *Applied Mathematical Sciences* **1**(8), 355–361.

Appendix A

Additional results - Hamiltonian p-median problem

Contents

This chapter presents additional results obtained by the branch-and-cut algorithm based on the PQR formulation discussed in Chapter 7. No analysis of the results is performed here. Instead, each subsequent section will refer to the point in Chapter 7 to which the results correspond to.

A.1 Results for asymmetric instances

This section presents additional results with respect to asymmetric instances and are a complement to the results presented in Section 7.6.1.

Table A.1: Optimal solution results for asymmetric instances (appendix)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
ftv33	8	5	1201	0	0	109	4
		10	1187	0	0	0	10
		15	1261	0	0	0	6
ftv35	8	5	1387	0	0	114	0
		10	1383	0	0	0	6
		15	1480	0	0	0	20
ftv38	8	5	1444	0	0	56	0
		10	1440	0	0	0	10
		15	1534	0	0	0	44
p43	16	5	199	7	1622	1446	52
		10	158	3	560	1029	57
		15	148	0	0	0	0
		20	160	0	0	0	11
ftv44	9	5	1543	1	31	275	15
		10	1522	0	0	0	0
		15	1573	0	2	1	36
		20	1691	0	0	0	0
ftv47	11	5	1672	1	2	128	8
		10	1652	0	0	0	0
		15	1703	1	1	0	90
		20	1815	0	0	0	0
ry48p	20	5	13497	1	90	417	23
		10	12868	1	21	165	10
		15	12677	1	26	120	16
		20	12517	0	0	0	0
ft53	8	5	6022	1	2	66	1
		10	5942	0	0	0	0
		15	6049	1	0	0	67
		20	6305	0	0	0	72
		25	7639	1	0	0	134
ftv55	10	5	1482	1	22	354	14
		10	1435	0	0	0	0

Continues on the next page

Table A.1

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
		15	1445	0	0	0	87
		20	1548	1	0	0	106
		25	1790	0	0	0	242
ftv64	9	5	1732	0	0	98	4
		10	1721	0	0	0	0
		15	1721	0	0	0	0
		20	1767	0	6	0	128
		25	1888	0	0	0	101
		30	2140	1	0	0	129

Table A.2: Linear programming relaxation results for asymmetric instances (appendix)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
ftv33	8	5	1201	1193.11	0.66	0	98	4
		10	1187	1186	0.08	0	4	14
		15	1261	1247.8	1.05	0	0	14
ftv35	8	5	1387	1383.75	0.23	0	77	1
		10	1383	1382.5	0.04	0	0	6
		15	1480	1471.25	0.59	0	0	12
ftv38	8	5	1444	1440.75	0.23	0	56	2
		10	1440	1439.5	0.03	0	0	10
		15	1534	1520.5	0.88	0	0	29
p43	16	5	199	187.833	5.61	0	108	1
		10	158	152	3.80	0	262	4
		15	148	148	0.00	0	125	0
		20	160	148	7.50	0	5	4
ftv44	9	5	1543	1529.25	0.89	0	32	3
		10	1522	1522	0.00	0	0	8
		15	1573	1570.5	0.16	0	0	35
		20	1691	1691	0.00	0	0	40
ftv47	11	5	1672	1661.13	0.65	0	95	0
		10	1652	1652	0.00	0	20	7
		15	1703	1691.5	0.68	0	0	55
		20	1815	1815	0.00	0	0	48
ry48p	20	5	13497	13165.3	2.46	0	125	1
		10	12868	12735.8	1.03	0	85	1
		15	12677	12579.3	0.77	0	72	0

Continues on the next page

Table A.2

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
		20	12517	12517	0.00	0	8	0
ft53	8	5	6022	6004.07	0.30	0	57	0
		10	5942	5942	0.00	0	2	41
		15	6049	6027.75	0.35	0	0	57
		20	6305	6272.3	0.52	0	0	116
		25	7639	6973.5	8.71	0	0	43
ftv55	10	5	1482	1460.33	1.46	0	220	2
		10	1435	1435	0.00	0	65	27
		15	1445	1443.5	0.10	0	0	90
		20	1548	1510	2.45	0	0	92
		25	1790	1754	2.01	0	0	232
ftv64	9	5	1732	1729.17	0.16	0	134	14
		10	1721	1721	0.00	0	131	58
		15	1721	1721	0.00	0	34	54
		20	1767	1760.5	0.37	0	0	146
		25	1888	1878	0.53	0	0	136
		30	2140	2098.5	1.94	0	0	206

A.2 Results for symmetric instances

This section presents additional results with respect to symmetric instances and are a complement to the results presented in Section 7.6.2.

Table A.3: Optimal results for symmetric instances (appendix)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
dantzig42	20	5	604	4	676	1977	17
		10	573	3	243	1033	11
		15	548	1	21	333	4
		20	532	0	0	0	0
swiss42	20	5	1155	1	52	294	10
		10	1084	2	226	1103	25
		15	1034	1	11	77	3
		20	1009	0	0	0	0
att48	22	5	29816	4	325	1442	44
		10	27456	2	92	758	50
		15	27009	2	192	800	42
		20	26692	1	15	170	4
gr48	23	5	4544	1	2	190	0
		10	4318	1	43	471	26
		15	4231	4	154	1622	58
		20	4157	0	10	132	15
hk48	18	5	10834	3	231	1804	57
		10	10345	6	514	3634	59
		15	9946	0	4	223	6
		20	9916	1	22	235	21
eil51	23	5	441	1	9	123	2
		10	428	2	83	652	7
		15	418	2	92	1124	58
		20	408	0	0	0	0
		25	409	0	0	2	0
berlin52	23	5	7052	3	125	1069	33
		10	6609	9	591	3113	114
		15	6444	2	60	887	15
		20	6359	2	79	580	9
		25	6373	0	0	2	0
brazil58	27	5	20150	14	1037	2135	109
		10	18407	61	3190	8346	227

Continues on the next page

Table A.3

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
		15	17582	120	5037	14661	330
		20	17017	53	2498	6665	151
		25	16583	0	0	88	4
st70	31	5	665	87	2949	9800	474
		10	631	156	6129	8656	300
		15	607	447	5972	20353	838
		20	589	435	7063	13435	356
		25	573	68	1538	3736	71
		30	561	1	6	374	1
		35	610	1	0	0	0
eil76	35	5	563	4	70	480	2
		10	550	4	55	923	23
		15	545	45	543	5528	735
		20	539	3	12	621	7
		25	536	5	60	1005	57
		30	533	4	40	732	34
		35	531	0	0	0	0
gr96	45	5	150721	45	833	2012	70
		10	147495	226	3205	10814	435
		15	144249	137	902	10476	475
		20	142035	492	2883	17145	815
		25	139977	560	2569	20101	1163
		30	138216	60	429	6985	137
		35	137453	222	1001	11954	375
		40	136338	23	214	3225	38
		45	135563	0	0	0	0
kroA100	45	5	20224	61	1214	2127	55
		10	19392	249	3805	11519	297
		15	18755	447	2479	22261	598
		20	18383	1481	5578	30894	529
		25	17924	113	456	7391	101
		30	17666	5801	3242	64283	785
		35	17432	287	1527	14109	111
		40	17212	8	68	1016	17
		45	17153	0	0	0	0
		50	18618	1	0	0	0

Continues on the next page

Table A.3

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
kroD100	44	5	20284	10221	60004	60366	7862
		10	[19106, 19372]	10800*	42459	103038	4750
		15	[18330, 18713]	10800*	27900	83788	5871
		20	17914	4433	12750	54159	3153
		25	17401	3989	5811	57340	2106
		30	17015	130	985	10774	231
		35	16752	4	4	537	5
		40	16650	4	18	419	12
		45	16625	1	0	16	3
		50	18474	39	668	0	0
kroE100	45	5	20839	88	1608	4251	189
		10	19595	16	98	1030	24
		15	18958	174	1342	11682	280
		20	18424	631	2963	27384	573
		25	17958	1556	3928	37284	809
		30	17489	82	546	6066	68
		35	17080	6	15	723	3
		40	16952	39	282	4674	33
		45	16741	0	0	0	0
		50	17730	1	0	0	0
rd100	46	5	7668	526	8975	11318	349
		10	7436	5829	38013	54921	3168
		15	[7211, 7234]	10800*	43503	98725	1938
		20	7028	8312	37825	84520	651
		25	6871	2506	7401	57975	1689
		30	6777	822	3463	21619	309
		35	6698	936	2119	33427	488
		40	6660	114	757	7620	87
		45	6617	5	25	1377	3
		50	6910	1	0	0	0

*Not solved to optimality within the limit of three hours.

Table A.4: Linear programming relaxation results for symmetric instances (appendix)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
dantzig42	20	5	604	579.5	4.06	0	131	0
		10	573	545	4.89	0	24	0

Continues on the next page

Table A.4

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
		15	548	534	2.55	0	79	0
		20	532	532	0.00	0	3	1
swiss42	20	5	1155	1123.17	2.76	0	83	0
		10	1084	1045	3.60	0	26	1
		15	1034	1021.83	1.18	0	62	2
		20	1009	1009	0.00	0	100	0
att48	22	5	29816	28724.3	3.66	0	227	0
		10	27456	27039.8	1.52	0	114	2
		15	27009	26675.5	1.23	0	33	1
		20	26692	26600.3	0.34	0	213	1
gr48	23	5	4544	4469.25	1.65	0	113	0
		10	4318	4248	1.62	0	23	0
		15	4231	4164.75	1.57	0	65	1
		20	4157	4139.6	0.42	0	102	2
hk48	18	5	10834	10511.8	2.97	0	107	0
		10	10345	10083.5	2.53	0	126	0
		15	9946	9899	0.47	0	204	0
		20	9916	9870	0.46	0	96	8
eil51	23	5	441	437.13	0.88	0	157	0
		10	428	423.9	0.96	0	101	0
		15	418	414.75	0.78	0	71	0
		20	408	408	0.00	0	222	1
		25	409	408	0.24	0	2	4
berlin52	23	5	7052	6816.5	3.34	0	206	0
		10	6609	6491.83	1.77	0	106	0
		15	6444	6388.75	0.86	0	244	0
		20	6359	6322.39	0.58	0	221	1
		25	6373	6312	0.96	0	3	0
brazil58	27	5	20150	18569.1	7.85	0	68	1
		10	18407	17369	5.64	0	88	0
		15	17582	16877	4.01	0	107	0
		20	17017	16652	2.14	0	50	0
		25	16583	16573	0.06	1	287	0
st70	31	5	665	643.25	3.27	1	268	0
		10	631	603.083	4.42	0	101	2
		15	607	574.25	5.40	1	239	2

Continues on the next page

Table A.4

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
		20	589	564	4.24	0	75	2
		25	573	561.5	2.01	2	291	0
		30	561	560	0.18	1	302	1
		35	610	560	8.20	0	0	0
eil76	35	5	563	560.1	0.52	0	121	1
		10	550	546.25	0.68	1	376	0
		15	545	539.75	0.96	0	109	2
		20	539	535.75	0.60	0	84	7
		25	536	533.167	0.53	1	205	0
		30	533	531.778	0.23	1	125	3
		35	531	531	0.00	0	20	0
gr96	45	5	150721	147533	2.12	1	240	0
		10	147495	143407	2.77	1	173	0
		15	144249	140685	2.47	2	393	0
		20	142035	138559	2.45	1	225	0
		25	139977	137118	2.04	1	158	1
		30	138216	136510	1.23	0	62	2
		35	137453	136096	0.99	2	162	0
		40	136338	135777	0.41	1	67	0
		45	135563	135563	0.00	1	75	0
kroA100	45	5	20224	19658	2.80	1	143	0
		10	19392	18699.5	3.57	1	226	1
		15	18755	18078	3.61	1	140	3
		20	18383	17726.9	3.57	1	146	0
		25	17924	17442.7	2.69	1	120	1
		30	17666	17278.3	2.19	2	233	0
		35	17432	17199.3	1.33	1	95	2
		40	17212	17168.5	0.25	3	177	0
		45	17153	17153	0.00	3	299	0
		50	18618	17153	7.87	0	0	0
kroD100	44	5	20284	19322	4.74	1	221	0
		10	[19106, 19372]	18353.7	5.26	1	170	0
		15	[18330, 18713]	17662.2	5.62	1	293	0
		20	17914	17177.8	4.11	1	125	0
		25	17401	16899.9	2.88	1	196	1
		30	17015	16740.5	1.61	1	156	1

Continues on the next page

Table A.4

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
		35	16752	16663	0.53	3	271	1
		40	16650	16617.3	0.20	24	828	0
		45	16625	16585	0.24	1	56	4
		50	18474	16585	10.23	0	0	0
kroE100	45	5	20839	20337.3	2.41	1	288	0
		10	19595	19096	2.55	1	126	0
		15	18958	18260.5	3.68	1	226	0
		20	18424	17763.5	3.58	1	100	0
		25	17958	17376.3	3.24	1	147	0
		30	17489	17151.8	1.93	4	257	0
		35	17080	16968.2	0.65	3	238	5
		40	16952	16803	0.88	1	108	0
		45	16741	16741	0.00	1	0	0
		50	17730	16741	5.58	0	0	0
rd100	46	5	7668	7368	3.91	1	184	0
		10	7436	7074.5	4.86	1	158	0
		15	[7211, 7234]	6904.67	4.55	4	611	0
		20	7028	6777.25	3.57	2	323	1
		25	6871	6698	2.52	2	225	1
		30	6777	6650	1.87	4	460	1
		35	6698	6625.75	1.08	10	670	0
		40	6660	6616.5	0.65	3	306	0
		45	6617	6613	0.06	8	471	8
		50	6910	6613	4.30	0	0	0

A.3 Numerical comparison between the x-v formulation and the PQR formulation

This section presents additional results with respect to the comparison between the x-v formulation and the PQR formulation for both asymmetric and symmetric instances and are a complement to the results presented in Section 7.7.1.

Table A.5: Numerical comparison between the x-v and the PQR formulations (appendix)

			B&C PQR			B&C x-v		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
ftv33	8	5	1193.11	1201	0	1194	1201	0
		10	1186	1187	0	1185	1187	0
		15	1247.8	1261	0	1209.63	1261	15
ftv35	8	5	1383.75	1387	0	1385	1387	0
		10	1382.5	1383	0	1381.5	1383	1
		15	1471.25	1480	0	1415.72	1480	59
ftv38	8	5	1440.75	1444	0	1442	1444	0
		10	1439.5	1440	0	1438.5	1440	0
		15	1520.5	1534	0	1465.05	1534	3775
p43	16	5	187.833	199	7	199	199	0
		10	152	158	3	157	158	0
		15	148	148	0	148	148	0
		20	148	160	0	148	160	138
ftv44	9	5	1529.25	1543	1	1529.89	1543	1
		10	1522	1522	0	1522	1522	0
		15	1570.5	1573	0	1540.83	1573	91
		20	1691	1691	0	1597.39	1691	3711
ftv47	11	5	1661.13	1672	1	1664.75	1672	1
		10	1652	1652	0	1652	1652	0
		15	1691.5	1703	1	1656	1703	453
		20	1815	1815	0	1707.81	[1774, 1837]	10800*
ry48p	20	5	13165.3	13497	1	13320	13497	1
		10	12735.8	12868	1	12797.8	12868	1
		15	12579.3	12677	0	12597.5	12677	0
		20	12517	12517	0	12517	12517	0
ft53	8	5	6004.07	6022	1	6012.8	6022	0
		10	5942	5942	0	5936.33	5942	5
		15	6027.75	6049	1	5960	6049	80

Continues on the next page

Table A.5

			B&C PQR			B&C x-v		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
		20	6272.3	6305	0	6031	[6242, 6305]	10800*
		25	6973.5	7639	1	6183.19	[6203, 8206]	10800*
ftv55	10	5	1460.33	1482	1	1462.67	1482	1
		10	1435	1435	0	1435	1435	0
		15	1443.5	1445	0	1438.33	1445	9
		20	1510	1548	1	1453.67	[1478, 1601]	10800*
		25	1754	1790	0	1515.98	[1436, 1802]	10800*
ftv64	9	5	1729.17	1732	0	1731.2	1732	0
		10	1721	1721	0	1721	1721	0
		15	1721	1721	0	1721	1721	0
		20	1760.5	1767	0	1726.83	[1735, 1843]	10800*
		25	1878	1888	0	1767.62	[1778, 1958]	10800*
		30	2098.5	2140	1	1836.25	[1721, 2229]	10800*
ft70	10	5	38055.6	38120	3	38113.7	38120	1
		10	37978	37978	0	37978	37978	0
		15	38018.5	38033	1	37984.5	38033	1078
		20	38275.2	38390	7	38013	[38040, 39247]	10800*
		25	39027.9	39233	9	38109.8	[38139, 40027]	10800*
		30	40258.3	40539	11	38356.7	[38394, 40823]	10800*
		35	42297	42908	2	38718.1	[38791, 44946]	10800*
dantzig42	20	5	579.5	604	4	599	604	0
		10	545	573	3	561.5	573	3
		15	534	548	1	542.7	548	1
		20	532	532	0	532	532	0
swiss42	20	5	1123.17	1155	1	1141.75	1155	1
		10	1045	1084	2	1065.5	1084	1
		15	1021.83	1034	1	1026.5	1034	0
		20	1009	1009	0	1009	1009	0
att48	22	5	28724.3	29816	4	29195.5	29816	1
		10	27039.8	27456	2	27256.5	27456	1
		15	26675.5	27009	2	26794.3	27009	1
		20	26600.3	26692	1	26610.5	26692	0
gr48	23	5	4469.25	4544	1	4526.06	4544	0
		10	4248	4318	1	4260.5	4318	1
		15	4164.75	4231	4	4183.17	4231	2

Continues on the next page

Table A.5

			B&C PQR			B&C x-v		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
		20	4139.6	4157	0	4140.5	4157	1
hk48	18	5	10511.8	10834	3	10769.6	10834	0
		10	10083.5	10345	6	10219.7	10345	1
		15	9899	9946	0	9918.8	9946	0
		20	9870	9916	1	9870	9916	2
eil51	23	5	437.13	441	1	438.571	441	1
		10	423.9	428	2	425.5	428	3
		15	414.75	418	2	414.8	418	2
		20	408	408	0	408	408	0
		25	408	409	0	408	409	0
berlin52	23	5	6816.5	7052	3	6913.43	7052	1
		10	6491.83	6609	9	6564	6609	2
		15	6388.75	6444	2	6400.5	6444	2
		20	6322.39	6359	2	6337	6359	1
		25	6312	6373	0	6312	6373	46
brazil58	27	5	18569.1	20150	14	19103.3	20150	1
		10	17369	18407	61	17776.5	18407	2
		15	16877	17582	120	17191.5	17582	2
		20	16652	17017	53	16748.5	17017	2
		25	16573	16583	0	16582	16583	0
st70	31	5	643.25	665	87	662.1	665	2
		10	603.083	631	156	621.4	631	26
		15	574.25	607	447	594.154	607	479
		20	564	589	435	576.3	589	29
		25	561.5	573	68	563.5	573	2
		30	560	561	1	560.167	561	0
		35	560	610	1	560	[560, 728]	10800*
eil76	35	5	560.1	563	4	560.889	563	6
		10	546.25	550	4	548.382	550	5
		15	539.75	545	45	543	545	4
		20	535.75	539	3	538	539	6
		25	533.167	536	5	534	536	4
		30	531.778	533	4	532	533	7
		35	531	531	0	531	531	0
pr76	36	5	91255.6	[96104, 97764]	10800*	93076.5	[96803, 97450]	10800*

Continues on the next page

Table A.5

			B&C PQR			B&C x-v		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
		10	85634	[90747, 91883]	10800*	88006	[90206, 92329]	10800*
		15	81211.5	86380	1066	84413.3	86380	51
		20	79365.2	82311	99	81247	82311	5
		25	78332	82040	1757	79681.2	[81148, 82253]	10800*
		30	77731.8	[80612, 81961]	10800*	78302.5	[79655, 82651]	10800*
		35	77207.3	77973	1	77260.5	77973	1
gr96	45	5	147533	150721	45	149292	150721	75
		10	143407	147495	226	145200	[146647, 147635]	10800*
		15	140685	144249	137	142426	[143749, 146342]	10800*
		20	138559	142035	492	140240	142035	9113
		25	137118	139977	560	138444	139977	64
		30	136510	138216	60	137192	138216	40
		35	136096	137453	222	136350	137453	37
		40	135777	136338	23	135861	136338	11
		45	135563	135563	0	135563	135563	0
rat99	45	5	1228.5	1237	8	1231.5	1237	3
		10	1196.5	1212	68	1207.6	1212	6
		15	1178.17	1195	39	1190.88	1195	48
		20	1164.17	1184	172	1175	1184	2837
		25	1154.38	1170	177	1161	1170	683
		30	1148	1159	41	1152.54	1159	48
		35	1144.44	1153	50	1147.86	[1151, 1159]	10800*
		40	1142.17	1145	15	1144	1145	10
		45	1142	1142	0	1142	1142	0
kroA100	45	5	19658	20224	61	20046.3	20224	148
		10	18699.5	19392	249	19236.3	19392	69
		15	18078	18755	447	18572.6	18755	94
		20	17726.9	18383	1481	18043.8	18383	75
		25	17442.7	17924	113	17717.8	17924	29
		30	17278.3	17666	5801	17451.6	17666	30
		35	17199.3	17432	287	17253.2	17432	27
		40	17168.5	17212	8	17179.5	17212	5
		45	17153	17153	0	17153	17153	0
		50	17153	18618	1	17153	[17153, 22460]	10800*
kroD100	44	5	19322	20284	10221	19915.5	20284	22

Continues on the next page

Table A.5

			B&C PQR			B&C x-v		
Name	p'	p	LP	OPT	t (s)	LP	OPT	t (s)
		10	18353.7	[19106, 19372]	10800*	18934.2	19275	150
		15	17662.2	[18330, 18713]	10800*	18234.2	[18489, 19128]	10800*
		20	17177.8	17914	4433	17670.4	17914	89
		25	16899.9	17401	3989	17259.2	17401	17
		30	16740.5	17015	130	16935.5	17015	11
		35	16663	16752	4	16718.5	16752	2
		40	16617.3	16650	4	16634	16650	5
		45	16585	16625	1	16585	16625	16
		50	16585	18474	39	16585	[16585, 20732]	10800*
kroE100	45	5	20337.3	20839	88	20427	20839	128
		10	19096	19595	16	19430	19595	21
		15	18260.5	18958	174	18644	18958	50
		20	17763.5	18424	631	18043.5	18424	1887
		25	17376.3	17958	1556	17629	17958	3899
		30	17151.8	17489	82	17292.9	17489	108
		35	16968.2	17080	6	17014.8	17080	7
		40	16803	16952	39	16827.1	16952	17
		45	16741	16741	0	16741	16741	0
		50	16741	17730	1	16741	[17062, 18524]	10800*
rd100	46	5	7368	7668	526	7508.33	[7550, 8143]	10800*
		10	7074.5	7436	5829	7222.75	[7361, 7716]	10800*
		15	6904.67	[7211, 7234]	10800*	7024.45	[7146, 7264]	10800*
		20	6777.25	7028	8312	6860	[6949, 7048]	10800*
		25	6698	6871	2506	6762.5	6871	64
		30	6650	6777	822	6703.75	6777	39
		35	6625.75	6698	936	6658.5	6698	27
		40	6616.5	6660	114	6629.75	6660	15
		45	6613	6617	5	6614	6617	1
		50	6613	6910	1	6613	[6613, 8100]	10800*

*Not solved to optimality within the time limit of three hours

A.4 Results for the variant in which two-node circuits are not allowed

This section presents additional results with respect to the variant of the Hamiltonian p-median problem in which two-node circuits are not allowed and are a complement to the results presented in Section 7.7.2.

Table A.6: Optimal solution results for symmetric instances (two-node circuits not allowed) (appendix)

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
dantzig42	8	3	648	1	2	226	5
		10	654	0	0	62	18
swiss42	7	4	1232	1	4	211	7
		6	1231	1	7	176	25
		8	1231	1	0	87	65
		10	1238	1	1	95	105
		14	1292	1	0	10	285
att48	5	4	31903.3	1	0	168	3
		6	31836.1	1	0	56	61
		9	32195.5	2	4	99	170
		12	32742.9	4	11	81	416
		16	[35667.8, 37874.3]	10800*	18923	989	29397
gr48	6	4	4841	3	124	417	120
		6	4805	2	57	308	243
		9	4926	13	674	364	1094
		12	5011	8	73	145	1094
		16	5445	208	1777	407	6848
hk48	6	4	11271	2	69	400	52
		6	11197	0	0	0	0
		9	11292	2	17	151	270
		12	11450	4	37	315	383
		16	12215	118	1834	558	4030
eil51	3	5	422.323	4	69	286	354
		7	424.356	5	101	153	540
		10	432.489	10	246	282	1114
		12	436.587	14	217	440	1518
		17	473.977	1701	11238	978	9965
berlin52	7	5	7182.23	2	88	282	46
		7	7167.2	1	0	134	14

Continues on the next page

Table A.6

Name	p'	p	OPT	t (s)	B&B	$\#(\leq p)$	$\#(\geq p)$
		10	7206.7	2	3	116	70
		13	7298.63	2	7	109	209
		17	7800.77	36	548	179	2031
brazil58	12	5	21744	12	676	1314	191
		8	21289	6	235	896	98
		11	21080	5	7	892	16
		14	21221	2	0	70	90
		19	22635	71	1135	556	2925
st70	12	7	638.221	5	25	437	18
		10	632.54	4	18	430	23
		14	630.902	3	0	102	141
		17	636.194	9	22	283	403
		23	694.495	3739	19720	2440	9607
pr76	8	7	101401	5	31	851	28
		10	101779	8	36	280	142
		15	103663	34	767	1583	306
		19	104482	7	6	123	210
		25	110074	12	0	174	385

*Not solved to optimality within the time limit of three hours

Table A.7: Linear programming relaxation results for symmetric instances (two-node circuits not allowed) (appendix)

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
dantzig42	8	3	648	641	1.08	0	79	0
		10	654	651.5	0.38	0	79	17
swiss42	7	4	1232	1214.5	1.42	0	197	0
		6	1231	1214.5	1.34	0	165	21
		8	1231	1218.81	0.99	0	131	43
		10	1238	1225.75	0.99	0	246	75
		14	1292	1270	1.70	0	22	104
att48	5	4	31903.3	31703.7	0.63	0	231	4
		6	31836.1	31671.4	0.52	0	134	55
		9	32195.5	31756.7	1.36	0	96	83
		12	32742.9	32083.2	2.01	1	195	188
		16	[35667.8, 37874.3]	33217	12.30	1	111	2892
gr48	6	4	4841	4770	1.47	0	125	1

Continues on the next page

Table A.7

Name	p'	p	OPT	LP	gap (%)	t_L (s)	$\#(\leq p)$	$\#(\geq p)$
		6	4805	4769.75	0.73	0	144	30
		9	4926	4807	2.42	0	37	224
		12	5011	4886.21	2.49	1	73	395
		16	5445	5119.74	5.97	4	234	711
hk48	6	4	11271	11197.5	0.65	0	174	2
		6	11197	11197	0.00	0	129	6
		9	11292	11250.3	0.37	0	180	139
		12	11450	11367.7	0.72	1	191	238
		16	12215	11742.7	3.87	1	92	378
eil51	3	5	422.323	419.58	0.65	1	131	155
		7	424.356	421.306	0.72	1	94	195
		10	432.489	424.737	1.79	1	128	312
		12	436.587	428.425	2.87	2	141	441
		17	473.977	445.355	6.04	3	379	423
berlin52	7	5	7182.23	7167.14	0.21	0	165	20
		7	7167.2	7166.87	0.00	0	141	21
		10	7206.7	7182.32	0.34	1	189	176
		13	7298.63	7263.34	0.48	1	247	225
		17	7800.77	7559.85	3.09	0	17	167
brazil58	12	5	21744	21001	3.42	0	196	1
		8	21289	20904	1.81	1	421	8
		11	21080	20902.4	0.84	3	581	31
		14	21221	21023.2	0.93	2	689	97
		19	22635	21631.7	4.43	1	266	205
st70	12	7	638.221	631.417	1.07	1	227	4
		10	632.54	628.559	0.63	6	1077	5
		14	630.902	628.708	0.35	13	1109	187
		17	636.194	630.017	0.97	10	676	367
		23	694.495	648.67	6.60	7	560	266
pr76	8	7	101401	99028.9	2.34	1	168	37
		10	101779	99263	2.47	1	91	63
		15	103663	100667	2.89	1	155	154
		19	104482	102559	1.84	2	234	237
		25	110074	108023	1.86	4	324	166